CS702 MID TERM PAPERS SOLVED

Question: Prove $\log_a b \cdot \log_b c = \log_a c$

$$\log_a b \,.\, \log_b c = \log_a c$$

Proof

let us suppose that: $\log_a^b = s$ $\log_b^c = t$

$\Rightarrow a^s = b$ and $b^t = c$

$Now\ b^t = c,\quad (a^s)^t = c$

$a^{st} = c,\qquad \log_a^c = s \cdot t$

$s \cdot t = \log_a^c,\qquad \log_a^b \cdot \log_b^c = \log_a^c$ proved

Question: Prove $3^{\log_4 n} = n^{\log_4 3}$

$$3^{\log_4 n} = n^{\log_4 3}$$

Proof: $3^{\log_4^n} = n^{\log_4^3}$

$$\Leftrightarrow \quad \log_3(3^{\log_4^n}) = \log_3(n^{\log_4^3})$$

$$\Leftrightarrow \quad \log_4^n \cdot \log_3^3 = \log_4^3 \cdot \log_3^n$$

$$\Leftrightarrow \quad \log_4^n = \log_4^3 \cdot \log_3^n$$

$$\Leftrightarrow \quad \log_4^n = \log_4^n$$

Question: Asymptotic Notations Properties

- Categorize algorithms based on asymptotic growth rate e.g. linear, quadratic, polynomial, and exponential
- Ignore small constant and small inputs
- Estimate upper bound and lower bound on growth rate of time complexity function
- Describe running time of algorithm as n grows to ∞.
- Describes behavior of function within the limit.

Limitations

- Not always useful for analysis on fixed-size inputs.
- All results are for sufficiently large inputs.

Asymptotic Notations

Asymptotic Notations **Θ**, **O**, **Ω**, **o**, ω

- We use **Θ** to mean “order exactly”
- **O** to mean “order at most”
- **Ω** to mean “order at least”
- **o** to mean “tight upper bound”
- ω to mean “tight lower bound”

What is Complexity?

The level in difficulty in solving mathematically posed problems as measured by

- The time (time complexity)
- Number of steps or arithmetic operations (computational complexity)
- Memory space required (space complexity)

Major Factors in Algorithms Design

1. Correctness

An algorithm is said to be correct if

• For every input, it halts with correct output.

• An incorrect algorithm might not halt at all OR

• It might halt with an answer other than desired one.

• Correct algorithm solves a computational problem

2. Algorithm Efficiency

Measuring efficiency of an algorithm,

• Do its analysis i.e. growth rate.

• Compare efficiencies of different algorithms for the same problem.

Question:

Prove $n^2 + n = O(n^3)$

Proof:

- Here, we have $f(n) = n^2 + n$, and $g(n) = n^3$
- Notice that if $n \geq 1$, $n \leq n^3$ is clear.
- Also, notice that if $n \geq 1$, $n^2 \leq n^3$ is clear.
- **Side Note:** In general, if $a \leq b$, then $n^a \leq n^b$ whenever $n \geq 1$. This fact is used often in these types of proofs.
- Therefore,

$$n^2 + n \leq n^3 + n^3 = 2n^3$$

- We have just shown that

$$n^2 + n \leq 2n^3 \text{ for all } n \geq 1$$

- Thus, we have shown that $n^2 + n = O(n^3)$ (by definition of Big-O, with $n_0 = 1$, and $c = 2$.)

Question: Prove that $2n^2 \in O(n^3)$

Proof:

Assume that $f(n) = 2n^2$, and $g(n) = n^3$

Now we have to find the existence of c and n_0

$f(n) \leq c.g(n) \Rightarrow 2n^2 \leq c.n^3 \Rightarrow 2 \leq c.n$

if we take, $c = 1$ and $n_0 = 2$ OR

$c = 2$ and $n_0 = 1$ then

$2n^2 \leq c.n^3$

Hence $f(n) \in O(g(n))$, $c = 1$ and $n_0 = 2$

Questions: Prove that f(n) = O(g(n))
f(n) ϵ O(g(n)) Big-Oh Notation (O) N log n

If $f, g: N \rightarrow R^+$, then we can define Big-Oh as

For a given function $g(n) \geq 0$, denoted by $O(g(n))$ the set of functions,

$O(g(n)) = \{f(n)$: there exist positive constants c and n_o such that

$0 \leq f(n) \leq cg(n)$, for all $n \geq n_o\}$

$f(n) = O(g(n))$ means function $g(n)$ is an asymptotically upper bound for $f(n)$.

We may write $f(n) = O(g(n))$ OR $f(n) \in O(g(n))$

Intuitively:
Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.

Transpose Symmetry

Question: Prove that $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

Proof

Since $f(n) = O(g(n)) \Rightarrow$

$\exists$ constants $c > 0$ and $n_0 \in N$ such that

$$0 \leq f(n) \leq cg(n) \qquad \forall\, n \geq n_0$$

Dividing both side by c

$$0 \leq (1/c)f(n) \leq g(n) \qquad \forall\, n \geq n_0$$

Put $1/c = c'$

$$0 \leq c'f(n) \leq g(n) \qquad \forall\, n \geq n_0$$

Hence, $g(n) = \Omega(f(n))$

Question: Prove that $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

<u>Proof</u>

Since $f(n) = o(g(n)) \Rightarrow$

$\exists$ constants $c > 0$ and $n_0 \in N$ such that

$$0 \le f(n) < cg(n) \qquad \forall\ n \ge n_0$$

Dividing both side by c

$$0 \le (1/c)f(n) < g(n) \qquad \forall\ n \ge n_0$$

Put $1/c = c'$

$$0 \le c'f(n) < g(n) \qquad \forall\ n \ge n_0$$

Hence, $g(n) = \omega(f(n))$

Question: Big-Omega Notation (Ω)

Solution:

If f, g: $N \rightarrow R^+$, then we can define Big-Omega as

For a given function $g(n)$ denote by $\Omega(g(n))$ the set of functions,

$\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_o such that

$0 \leq cg(n) \leq f(n)$ for all $n \geq n_o\}$

$f(n) = \Omega(g(n))$, means that function $g(n)$ is an asymptotically lower bound for $f(n)$.

We may write $f(n) = \Omega(g(n))$ OR $f(n) \in \Omega(g(n))$

Intuitively:

Set of all functions whose *rate of growth* is the same as or higher than that of $g(n)$.

Question: Prove that $f(n) = O(g(n)) \leftrightarrow g(n) = \Omega(f(n))$ (Omega)

Solution:

Prove that $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

Proof

Since $f(n) = O(g(n)) \Rightarrow$

$\exists$ constants $c > 0$ and $n_0 \in N$ such that

$0 \leq f(n) \leq cg(n) \quad \forall\ n \geq n_0$

Dividing both side by c

$0 \leq (1/c)f(n) \leq g(n)\ \forall\ n \geq n_0$

Put $1/c = c'$

$0 \leq c'f(n) \leq g(n) \quad \forall\ n \geq n_0$

Hence, $g(n) = \Omega(f(n))$

[Little Omega]

Question: Prove that $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega\ f(n))$

Solution:

<u>Proof</u>

Since $f(n) = o(g(n)) \Rightarrow$

$\exists$ constants $c > 0$ and $n_0 \in N$ such that

$$0 \le f(n) < cg(n) \qquad \forall\ n \ge n_0$$

Dividing both side by c

$$0 \le (1/c)f(n) < g(n) \qquad \forall\ n \ge n_0$$

Put $1/c = c'$

$$0 \le c'f(n) < g(n) \qquad \forall\ n \ge n_0$$

Hence, $g(n) = \omega(f(n))$

Question: Prove that $5.n^2$ Є $\omega(n)$ OR [$5n^2$ Є $\omega(n)$]

<u>Proof:</u>

Assume that $f(n) = 5.n^2$, and $g(n) = n$

We have to prove that for any c there exists n_0 such that, $c.g(n) < f(n)$ for all $n \ge n_0$

$c.n < 5.n^2 \Rightarrow c < 5.n$

This is true for any c, because for any arbitrary c e.g. $c = 1000000$, we can choose $n_0 = 1000000/5 = 200000$ and the above inequality does hold.

And hence $f(n) \in \omega(g(n))$,

[Little Omega]

Question: Prove that $5.n + 10 \notin \omega(n)$ OR [$5n + 10 \notin \omega(n)$]

Proof:

Assume that $f(n) = 5.n + 10$, and $g(n) = n$

We have to find the existence n_0 for any c, s.t.

$c.g(n) < f(n)$ for all $n \geq n_0$

$c.n < 5.n + 10$, if we take $c = 16$ then

$16.n < 5.n + 10 \Leftrightarrow 11.n < 10$ is not true for any positive integer.

Hence $f(n) \notin \omega(g(n))$

Question: Prove that $100.n \notin \omega(n)$ OR [$100n \notin \omega(n)$]

Proof:

Let $f(n) = 100.n$, and $g(n) = n^2$

Assume that $f(n) \in \omega(g(n))$

Now if $f(n) \in \omega(g(n))$ then there n_0 for any c such that

$c.g(n) < f(n)$ for all $n \geq n_0$ this is true

$\Rightarrow c.n^2 < 100.n \Rightarrow c.n < 100$

If we take $c = 100$, $n < 1$, not possible

Hence $f(n) \notin \omega(g(n))$ i.e. $100.n \notin \omega(n^2)$

Question: For all integers a and b and any nonnegative integer n,

gcd(an, bn) = n gcd(a, b).

Proof

If n = 0, the corollary is trivial.

If n > 0, then gcd(an, bn) is the smallest positive element of the set {anx + bny},

i.e.

gcd(an, bn) = min {anx + bny}

= min{n.{ax + by}}

= n. min{ax + by}

n times smallest positive element of set {ax + by}.

Hence gcd(an, bn) = n.gcd(x, y)

Solving Modular Linear Equations

Question: A congruence of the form $ax \equiv b \pmod{m}$ is called a linear congruence.

Solving:

- To solve this congruence, objective is to find the x that satisfy the given equation.
- An inverse of a, modulo m is any integer a' such that, $a'a \equiv 1 \pmod{m}$.
- If we can find such an a', then we can solve $ax \equiv b$ by multiplying throughout by it, giving $a'ax \equiv a'b$,
- Thus, $1 \cdot x \equiv a'b$, $\Rightarrow x \equiv a'b \pmod{m}$.

Question: Proof by Induction Prove that $n^2 \geq n + 100 \quad \forall\, n \geq 11$

Solution

Let $P(n) * n^2 \geq n + 100 \qquad \forall n \geq 11$

1. $P(11) * 11^2 \geq 11 + 100 \qquad 121 \geq 111$, which is true

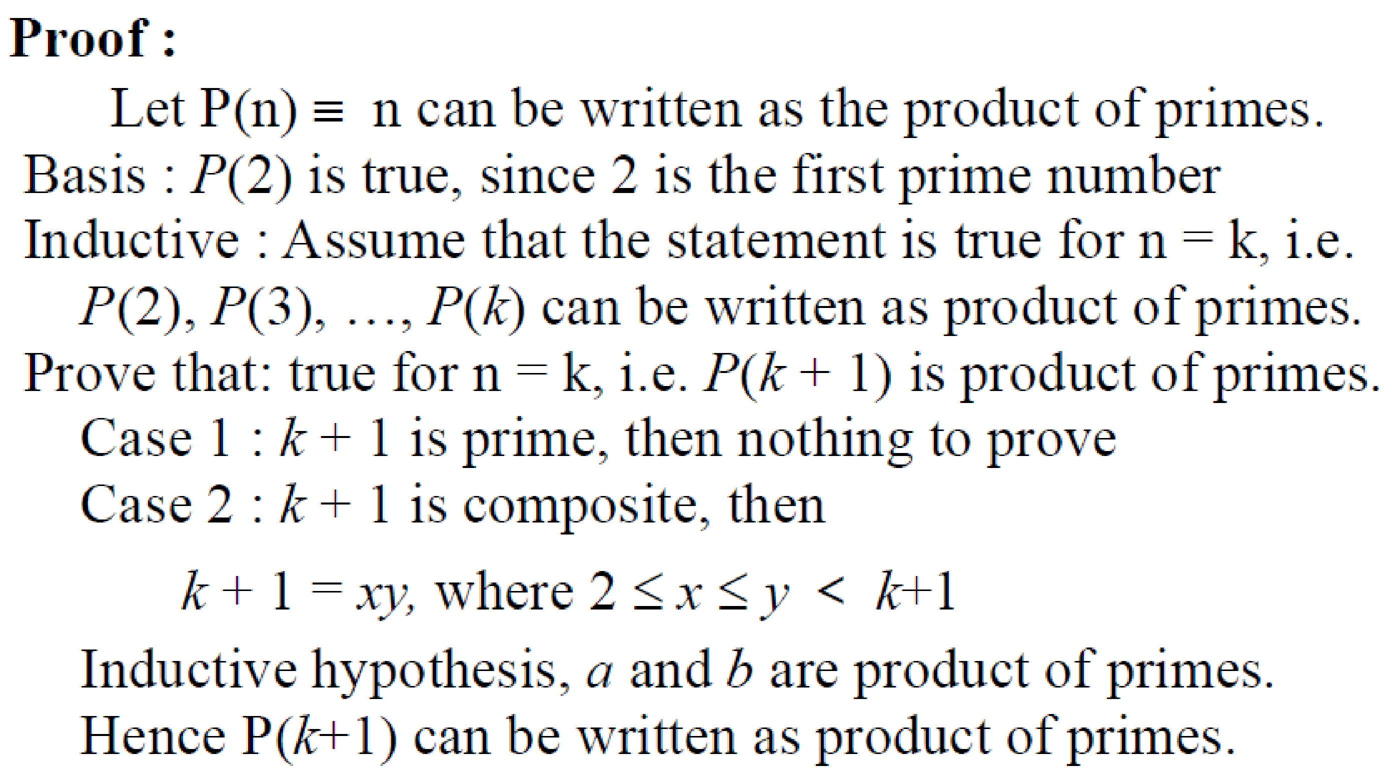
2. Suppose predicate is true for n = k, i.e.
$P(k) * k^2 \geq k + 100$, true $k \geq 11$

3. Now it can be proved that

$P(k+1) * (k+1)^2 \geq (k+1) + 100$,
$k^2 + 2k + 1 \geq k + 1 + 100$
$k^2 + k \geq 100$ (by 1 and 2)

Hence $P(k) \geq P(K+1)$

Question: If n Є Z, n >1, then n can be written as Product of Primes.



Proof :

Let P(n) ≡ n can be written as the product of primes.

Basis : $P(2)$ is true, since 2 is the first prime number

Inductive : Assume that the statement is true for n = k, i.e.

$P(2)$, $P(3)$, …, $P(k)$ can be written as product of primes.

Prove that: true for n = k, i.e. $P(k + 1)$ is product of primes.

Case 1 : $k + 1$ is prime, then nothing to prove

Case 2 : $k + 1$ is composite, then

$$k + 1 = xy, \text{ where } 2 \leq x \leq y < k+1$$

Inductive hypothesis, a and b are product of primes.

Hence P(k+1) can be written as product of primes.

Question: Property: Totient Function

Prove that $\varphi(p.q) = (p-1).(q-1)$, where p and q are prime numbers

Proof

If $n = p$, a prime number, then $\phi(p) = (p-1)$; e.g. ($\varphi(7) = 6$ because 7 is prime)

If $n = p * q$ where p and q are both prime then $\phi(n) = \phi(p*q)$

As above $\phi(p) = p - 1$

Similarly $\phi(q) = q - 1$

For $\phi(n) = \phi(p*q)$, the residues will be

$S_1 = \{0, 1, 2,. . ., (pq-1)\}$

Out of S_1, residues that are not relatively prime to n:

$S_2 = \{p, 2p, \dots(q-1)p\}$, $S_3 = \{q, 2q,\dots\dots(p-1)q\}$, $S_4 = \{0\}$

The number of elements of $S_1 = pq$

The number of elements of $S_2 = q-1$

The number of elements of $S_3 = p-1$

The number of elements of $S_4 = 1$

Hence number of relatively prime elements in S_1 is

$\phi(n) = pq – [(q-1)+(p-1)+1]$

$= pq – q + 1 – p + 1 -1$

$= pq – q – p + 1 = (p-1)(q-1) = \phi(p) * \phi(q)$

Lemma:

Question: For any integer n and any prime number p, if p divides n then p does not divide n + 1

Proof

- Express the statement in the form: $\forall x \in D, P(x) \Rightarrow Q(x)$
- Let, Z = set of all integers, and P = set of all primes
- $D(p, n) \equiv$ p divides n
- $DN(p, n) \equiv$ p does not divide n
- Now our problem becomes

$$\forall n \in Z, p \in P, D(p, n) \Rightarrow DN(p, n + 1)$$

- Suppose that for some integer n and prime p, p divides n $\equiv D(p, n)$
- Now we have to prove that p does not divide n + 1
- On contrary we suppose that p divide n + 1
- It means that there exists an integer q_1 such that $n + 1 = pq_1$
- Since p divides n. Hence there exists an integer q_2 such that $n = pq_2$
- Now, $n + 1 - n = pq_1 - pq_2$

  $1 = pq_1 - pq_2 = p(q_1 - q_2) \Rightarrow p = 1$ or -1 contradiction
- Hence p does not divide n + 1 $\equiv DN(p, n)$

Question: Lemma Statement:

The square of an odd integer is of the form 8m + 1 for some integer m.

- Suppose n is an arbitrary odd integer.
- By quotient remainder theorem any integer has the form
  4m, 4m + 1, 4m + 2 OR 4m+3
- Now since n is an odd integer, hence n can be represented as
  4m + 1 OR 4m+3
- Now we have to prove that squares of 4m + 1 and 4m + 3 are of the form 8m + 1.

Case 1

Square of 4m + 1

$(4m + 1)^2 = 16m^2 + 8m + 1 = 8(2m^2 + m) + 1$

$= 8m' + 1$, where $m' = (2m^2 + m)$

Case 2

Square of 4m + 3

$(4m + 3)^2 = 16m^2 + 24m + 9 = 8(2m^2 + 3m + 1) + 1$

$= 8m'' + 1$, where $m'' = (2m^2 + 3m + 1)$

- Hence any odd integer has the form 8m + 1 for some m

Question: Lemma : Statement

If n Є N, n > 1 is not prime then n is divisible by some prime number

p ≤ square root n i.e. p ≤ $\sqrt{n}$.

Proof

- Since n is not prime hence it can be factored as

  n = x.y where 1 <x ≤ y <n

- x or y is a prime, if not it can be further factored out.
- Also suppose without loss of generality that x ≤ y
- Now our claim is that x ≤ $\sqrt{n}$
- This is because otherwise x.y > n, a contradiction
- We only require to check till $\sqrt{n}$ for primality test.

Question: Theorem

Prove, by mathematical induction, that computational cost of generating permutations is n!.

Proof

- If n = 1, then the statement is true, because 1! =1
- If there are *k* elements in set then number of permutation = *k*!
- If we add one more element in any of the permutations, there will be *k*+1 number of ways to add it, resulting *k*+1 no. of permutations.
- Now total number of permutations = k!(k+1) = (k+1)!
- Hence true for all n.

Question: <u>Theorem</u>

Statement: If *a* and *b* are any integers, not both zero, then gcd(*a*, *b*) is the smallest positive element of the set {*ax* + *by* : *x*, *y* ∈ Z} of linear combinations of *a* and *b*.

Proof

Let s be the smallest positive such linear combination of a and b, i.e.

s = ax + by, for some x, y Є Z

By quotient remainder theorem

a = qs + r = qs + a mod s, where q = ⌊a/s⌋.

a mod s = a – qs = a - q(ax + by) = a (1 - qx) + b(-qy)

• Hence a ***mod*** s is a linear combination of a and b.

• But, since a mod s < s, therefore, a mod s = 0

• Now a mod s = 0 ⇒ s | a

• Similarly we can prove that, s | b.

• Thus, s is a common divisor of both a and b,

• Therefore, s ≤ gcd(a, b) (1)

• We know that if d | a and d | b then d | ax + by for all x, y integers.

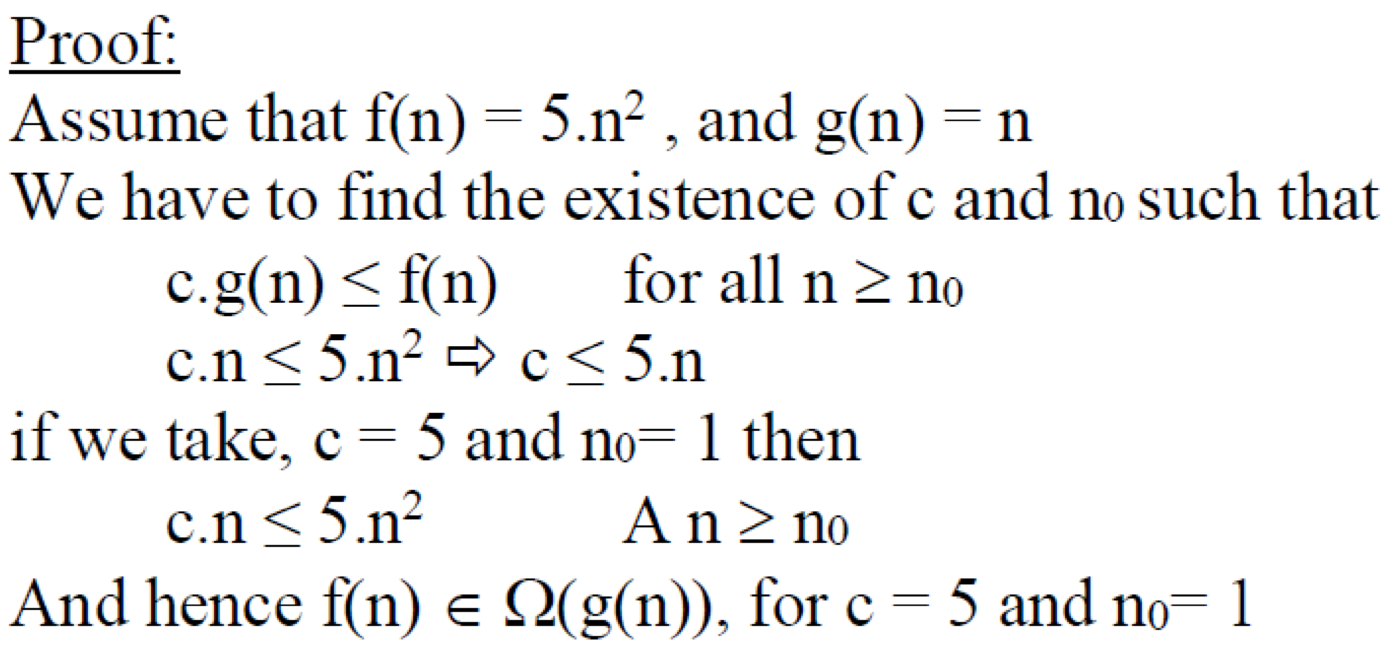
• Since gcd(a, b)| a and gcd(a, b) | b, hence gcd(a, b) | s and s > 0 imply that

gcd(a, b) ≤ s (2)

• By (1) and (2) proved that gcd(a, b) = s

Question: Prove that $5.n^2 \in \Omega(n)$ (Omega n)

Solution:



Proof:

Assume that $f(n) = 5.n^2$, and $g(n) = n$

We have to find the existence of c and n_0 such that

$c.g(n) \leq f(n)$ for all $n \geq n_0$

$c.n \leq 5.n^2 \Rightarrow c \leq 5.n$

if we take, $c = 5$ and $n_0 = 1$ then

$c.n \leq 5.n^2$ A $n \geq n_0$

And hence $f(n) \in \Omega(g(n))$, for $c = 5$ and $n_0 = 1$

Question: $n^3 + 4n^2 = \Omega(n^2)$ [Omega]

Proof:

- Here, we have $f(n) = n^3 + 4n^2$, and $g(n) = n^2$
- It is not too hard to see that if $n \geq 0$,

$$n^3 \leq n^3 + 4n^2$$

- We have already seen that if $n \geq 1$,

$$n^2 \leq n^3$$

- Thus when $n \geq 1$,

$$n^2 \leq n^3 \leq n^3 + 4n^2$$

- Therefore,

$$1n^2 \leq n^3 + 4n^2 \text{ for all } n \geq 1$$

- Thus, we have shown that $n^3 + 4n^2 = \Omega(n^2)$
(by definition of Big-Ω, with $n_0 = 1$, and $c = 1$.)

Question: Prove that 100.n + 5 $\notin \Omega(n^2)$ OR 100n + 5 $\notin \Omega(n^2)$ [Omega]

Proof:

Let $f(n) = 100.n + 5$, and $g(n) = n^2$

Assume that $f(n) \in \Omega(g(n))$

Now if $f(n) \in \Omega(g(n))$ then there exist c and n_0 such that

$c.g(n) \leq f(n)$ for all $n \geq n_0$

$c.n^2 \leq 100.n + 5$

$c.n \leq 100 + 5/n$

$n \leq 100/c$, for a very large n, which is not possible

And hence $f(n) \notin \Omega(g(n))$

Question: From asymptotic notation prove Theeta function / Theeta Notation (Θ)

Solution

Theta Notation (Θ)

If f, g: $N \rightarrow R^+$, then we can define Big-Theta as

For a given function $g(n)$ denoted by $\Theta(g(n))$ the set of functions,

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 and n_o such that

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_o\}$

$f(n) = \Theta(g(n))$ means function $f(n)$ is equal to $g(n)$ to within a constant factor, and $g(n)$ is an asymptotically tight bound for $f(n)$.

We may write $f(n) = \Theta(g(n))$ OR $f(n) \in \Theta(g(n))$

Intuitively: Set of all functions that have same *rate of growth* as g(*n*).

Question: Prove that $1/2.n^2 – 1/2.n = \Theta(n^2)$ OR [$1/2\ n^2 – 1/2n$] [Theeta]

Proof:

Assume that $f(n) = ½.n^2 – ½.n$, and $g(n) = n^2$

We have to find the existence of c_1, c_2 and n_0 such that

$c_1.g(n) \leq f(n) \leq c_2.g(n)$ for all $n \geq n_0$

Since, $½\ n^2 - ½\ n \leq ½\ n^2$ $\forall n \geq 0$ if $c_2 = ½$ and

$½\ n^2 - ½\ n \geq ½\ n^2 - ½\ n\ .\ ½\ n\ (\ \forall n \geq 2\ ) = ¼\ n^2,\ c_1 = ¼$

Hence $½\ n^2 - ½\ n \leq ½\ n^2 \leq ½\ n^2 - ½\ n$

$c_1.g(n) \leq f(n) \leq c_2.g(n)$ $\forall\ n \geq 2$, $c_1 = ¼$, $c_2 = ½$

Hence $f(n) \in \Theta(g(n)) \Rightarrow ½.n^2 – ½.n = \Theta(n^2)$

Theeta

Question: Prove that $a.n^2 + b.n + c = \Theta(n^2)$ OR [$an^2 + bn + c$]

Where a, b, c are constants and a > 0

Proof

If we take $c_1 = ¼.a$, $c_2 = 7/4.a$ and $n_0 = 2.\max((|b|/a), \sqrt{(|c|/a)})$

Then it can be easily verified that

$0 \leq c_1.g(n) \leq f(n) \leq c_2.g(n), \forall n \geq n_0$, $c_1 = ¼.a$, $c_2 = 7/4.a$

Hence $f(n) \in \Theta(g(n)) \Rightarrow a.n^2 + b.n + c = \Theta(n^2)$

Hence any polynomial of degree 2 is of order $\Theta(n^2)$

Theeta

Question: Prove that $2.n^2 + 3.n + 6 = \Theta(n^3)$ [$2n^2 + 3n + 6$]

Solution:

Proof:

Let $f(n) = 2.n^2 + 3.n + 6$, and $g(n) = n^3$

we have to show that f(n) ä $\Theta(g(n))$

On contrary assume that $f(n) \in \Theta(g(n))$ i.e.

there exist some positive constants c_1, c_2 and n_0 such that: $c_1.g(n) \leq f(n) \leq c_2.g(n)$

$c_1.g(n) \leq f(n) \leq c=.g(n) \Rightarrow c_1.n^3 \leq 2.n^2 + 3.n + 6 \leq c_2.\ n^3 \Rightarrow$

$c_1.n \leq 2 + 3/n + 6/n^2 \leq c_2.\ n \Rightarrow$

$c_1.n \leq 2 \leq c_2.\ n$, for large $n \Rightarrow$

$n \leq 2/c_1 \leq c_2/c_1.n$ which is not possible

Hence $f(n) = \Theta(g(n)) \Rightarrow 2.n^2 + 3.n + 6 = \Theta(n^3)$

Question: Prove that n^2 Є $O(n^2)$

Proof:

Assume that $f(n) = n^2$, and $g(n) = n^2$

Now we have to show that $f(n) \in O(g(n))$

Since

$f(n) \le c.g(n) \Rightarrow n^2 \le c.n^2 \Rightarrow 1 \le c$, take, $c = 1$, $n_0 = 1$

Then

$n^2 \le c.n^2$ for $c = 1$ and $n \ge 1$

Hence, $2n^2 \in O(n^2)$, where $c = 1$ and $n_0 = 1$

Question: Prove that $1000.n^2 + 1000.n$ Є $O(n^2)$

Proof:

Assume that $f(n) = 1000.n^2 + 1000.n$, and $g(n) = n^2$

We have to find existence of c and n_0 such that

$0 \le f(n) \le c.g(n)$ A $n \ge n_0$

$1000.n^2 + 1000.n \le c.n^2 = 1001.n^2$, for $c = 1001$

$1000.n^2 + 1000.n \le 1001.n^2$

$\Rightarrow 1000.n \le n^2 \Rightarrow n^2 \ge 1000.n \Rightarrow n^2 - 1000.n \ge 0$

$\Rightarrow n (n-1000) \ge 0$, this true for $n \ge 1000$

$f(n) \le c.g(n)$ A $n \ge n_0$ and $c = 1001$

Hence $f(n) \in O(g(n))$ for $c = 1001$ and $n_0 = 1000$

Question: Prove that n^3 Є $O(n^2)$

Proof:

On contrary we assume that there exist some positive constants c and n_0 such that

$0 \le n^3 \le c.n^2 \quad A\ n \ge n_0$

$0 \le n^3 \le c.n^2 \Rightarrow n \le c$

Since c is any fixed number and n is any arbitrary constant, therefore $n \le c$ is not possible in general.

Hence our supposition is wrong and $n^3 \le c.n^2$,

$A\ n \ge n_0$ is not true for any combination of c and n_0. And hence, $n^3 \in O(n^2)$

Question: Prove by contradiction that $g(n) = n^3$ is not $O(n^2)$

Assume that g(n) is $O(n^2)$. Then there are positive reals c and k such that

$| n^3 | \le c|n^2 |$, for all $n \ge k$.

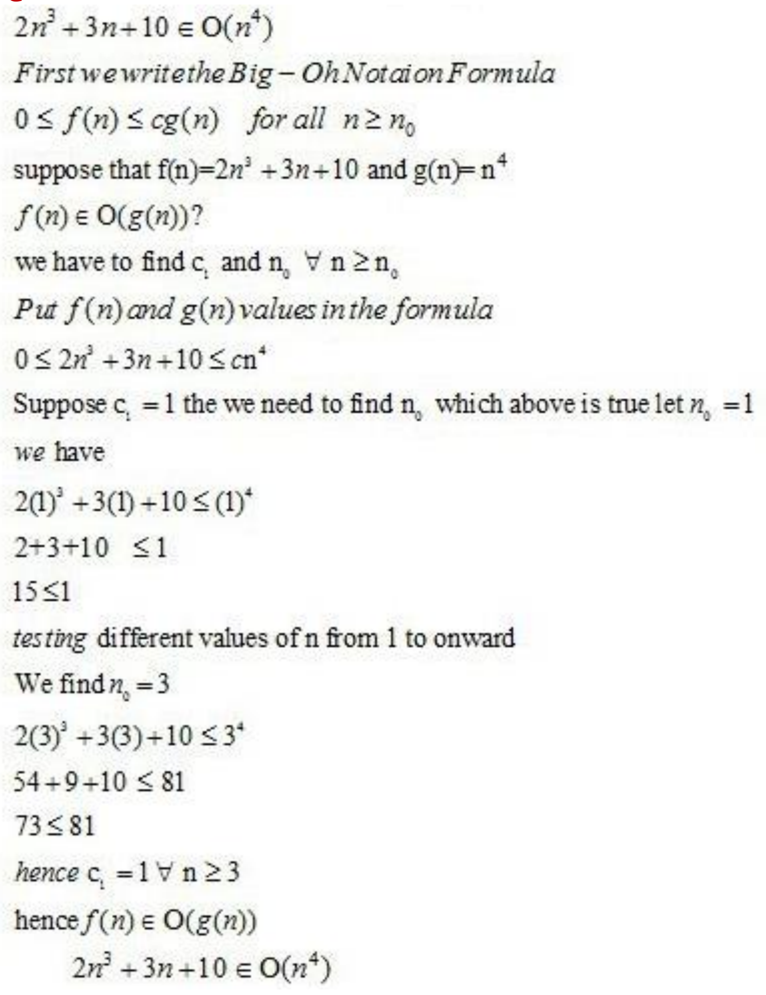
Without loss of generality, we can assume that n^2 is non-zero.

So then, dividing by n^2 we have $n \le c$.

But this is going to fail for those values of n that are bigger than c.

Contradiction! Thus, $g(n) = n^3$ is not $O(n^2)$.

Questions: Prove that $2.n^3 + 3.n + 10 \in O(n^4)$



$2n^3+3n+10 \in O(n^4)$

First we write the Big – Oh Notaion Formula

$0 \le f(n) \le cg(n)$ *for all* $n \ge n_0$

suppose that f(n)=$2n^3+3n+10$ and g(n)= n^4

$f(n) \in O(g(n))$?

we have to find c_1 and n_0 $\forall$ $n \ge n_0$

Put $f(n)$ *and* $g(n)$ *values in the formula*

$0 \le 2n^3+3n+10 \le cn^4$

Suppose c_1 $=1$ the we need to find n_0 which above is true let n_0 $=1$

we have

$2(1)^3+3(1)+10 \le (1)^4$

$2+3+10 \le 1$

$15 \le 1$

testing different values of n from 1 to onward

We find $n_0=3$

$2(3)^3+3(3)+10 \le 3^4$

$54+9+10 \le 81$

$73 \le 81$

hence c_1 $=1$ $\forall$ $n \ge 3$

hence $f(n) \in O(g(n))$

$2n^3+3n+10 \in O(n^4)$

Solution 2

$2n^3+3n+10 \in O(n^4)$

let suppose f(n)=$2n^3+3n+10$ and g(n)=n^4

$f(n) \in O(g(n))$?

we have to find c_1 and n_0 $\forall$ $n \ge n_0$

$0 \le 2n^3+3n+10 \le cn^4$

let c_1 $=1$ we need to n_0 for which above is true

testing different values from 1 to onward

$2(3)^3+3(3)+10 \le 3^4$

$73 \le 81$

hence c=1 for $\forall$ $n \ge 3$

hence $f(n) \in O(g(n)) = 2n^3+3n+10 \in O(n^4)$

c=1 for $\forall$ $n \ge 3$

Question: Prove that $5.n^2 + 10.n + 16 \in \Theta(n^2)$ (Theeta)

$$5n^2 + 10n + 16 \in \Theta(n^2)$$

Solution:

In order to prove the given Theta (Θ) notation, consider the following

Suppose

$f(n) = 5n^2 + 10n + 16$ and $g(n) = n^2$

On contrary assume that $f(n) \in \Theta(g(n))$ i.e. there are some positive constants c_1, c_2 and n_0 such that:
$c_1.g(n) \leq f(n) \leq c_2.g(n) \quad \forall\, n \geq n_0$

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

$\Rightarrow c_1 n^2 \leq 5n^2 + 10n + 16 \leq c_2 n^2$ [1]

Now, consider for $c_1 = 5$ and find n_0, which is true for the above

Let $n_0 = 1$, so we have

$5(1) \leq 5(1) + 10(1) + 16 \Rightarrow 5 \leq 31$

Hence $c_1 = 5 \quad \forall\, n \geq 1$

Now taking $c_2 = 6$, and find n_0, for this above [1] is true

$5n^2 + 10n + 16 \leq 6n^2$

For $n = 1 \Rightarrow 5 + 10 + 16 \leq 6$

$\Rightarrow 31 \leq 6$ [test is failed for n = 1]

For $n = 2 \Rightarrow 5(4) + 10(2) + 16 \leq 6(4)$

$\Rightarrow 56 \leq 24$ [test is failed for n = 2]

.

.

.

By testing the different values in [1] starting from 1 to onward, find that $n_0 = 12$ as

For $n = 12 \Rightarrow 5(144) + 10(12) + 16 \leq 6(144)$

$\Rightarrow 856 \leq 864$

Hence $c_2 = 6 \quad \forall\, n \geq 12$

Finally is proved that $f(n) \in \Theta(g(n))$, therefore $5n^2 + 10n + 16 \in \Theta(n^2)$

Question: Little-o Notation

o-Notation is used to denote a upper bound that is not asymptotically tight.

For a given function $g(n) \geq 0$, denoted by $o(g(n))$ the set of functions,

$$o(g(n)) = \left\{ \begin{array}{l} f(n): \text{for any positive constants } c, \text{ there exists a constant } n_o \\ \text{such that } 0 \leq f(n) < cg(n) \text{ for all } n \geq n_o \end{array} \right\}$$

f(n) becomes insignificant relative to g(n) as n approaches infinity

e.g., $2n = o(n^2)$ but $2n^2 \neq o(n^2)$.. $\lim\limits_{n \to \infty} \dfrac{f(n)}{g(n)} = 0$

g(n) is an upper bound for f(n), not asymptotically tight

Question: Prove that $2n^2 \in o(n^3)$ [Little o]

Proof:

Assume that $f(n) = 2n^2$, and $g(n) = n^3$

Now we have to find the existence n_0 for any c
f(n) < c.g(n) this is true
$\Rightarrow 2n^2 < c.n^3 \Rightarrow 2 < c.n$
This is true for any c, because for any arbitrary c we can choose n_0 such that the above inequality holds.

Hence $f(n) \in o(g(n))$

Little-o Notation

Question: Prove that $n^2 \notin o(n^2)$

Proof:

Assume that $f(n) = n^2$, and $g(n) = n^2$

Now we have to show that $f(n) \notin o(g(n))$

Since

$f(n) < c.g(n) \Rightarrow n^2 < c.n^2 \Rightarrow 1 \leq c,$

In our definition of small o, it was required to prove for any c but here there is a constraint over c. Hence, $n^2 \notin o(n^2)$, where $c = 1$ and $n_0 = 1$

Question: Prove that $1000.n^2 + 1000.n \notin o(n^2)$

Proof:

Assume that $f(n) = 1000.n^2 + 1000.n$, and $g(n) = n^2$

we have to show that $f(n) \notin o(g(n))$ i.e.

We assume that for any c there exist n_0 such that

$0 \leq f(n) < c.g(n)$ for all $n \geq n_0$

$1000.n^2 + 1000.n < c.n^2$

If we take $c = 2001$, then, $1000.n^2 + 1000.n < 2001.n^2$

$\Rightarrow 1000.n < 1001.n^2$ which is not true

Hence $f(n) \notin o(g(n))$ for $c = 2001$

Question: Let N be a set of natural numbers. The symbols, < (less than), ≤ (less than or equal) and = (equal) are relations over N. Prove or disprove the following.

1. < is reflexive, symmetric and transitive

2. ≤ is reflexive, symmetric and transitive

3. = is reflexive, symmetric and transitive

Solution:

1. < is reflexive, symmetric and transitive?

Reflexive:

A binary relation R on a set A is said to be reflexive if

(x, x) Є R, for all x Є A.

'<' Relationship is not reflexive.

Example:

x < x is not possible.

Symmetric:

A relation R between the elements in a universal set is called symmetric when x R y always implies y R x; in other words, when R is symmetric it does not matter whether x is written first or y is written first in the expression x R y.

'<' Relationship is not symmetric.

Example:

If x < y then y < x is not possible.

Transitive:

A relation R between the elements in a universal set is called transitive when x R y and y R z always imply x R z.

'<' is transitive.

Example:

If x < y and y < z; then x < z is true.

2. ≤ is reflexive, symmetric and transitive?

Reflexive:

A binary relation R on a set A is said to be reflexive if

(x, x) Є R, for all x Є A.

'≤' Relationship is reflexive.

Example:

$x \leq x$ is true.

Symmetric:

A relation R between the elements in a universal set is called symmetric when x R y always implies y R x; in other words, when R is symmetric it does not matter whether x is written first or y is written first in the expression x R y.

'≤' Relationship is not symmetric.

Example:

If $x \leq y$ then $y \leq x$ is not possible for $x < y$.

Transitive:

A relation R between the elements in a universal set is called transitive when x R y and y R z always imply x R z.

'≤' Relationship is transitive.

Example:

If $x \leq y$ and $y \leq z$; then $x \leq z$ is true.

3. = is reflexive, symmetric and transitive

Reflexive:

A binary relation R on a set A is said to be reflexive if

(x, x) Є R, for all x Є A.

‘=’ Relationship is reflexive.

Example:

x = x is true.

Symmetric:

A relation R between the elements in a universal set is called symmetric when x R y always implies y R x; in other words, when R is symmetric it does not matter whether x is written first or y is written first in the expression x R y.

‘=’ Relationship is symmetric.

Example:

If x = y then y = x is true.

Transitive:

A relation R between the elements in a universal set is called transitive when x R y and y R z always imply x R z.

‘=’is transitive.

Example:

If x = y and y = z then x = z is true.

Question: Use the logical equivalences above to show that **¬(p ∨ ¬(p ∧ q))** is a contradiction.

$$\neg (p \lor \neg (p \land q))$$

Solution.

$$\begin{aligned}
&\neg(p \lor \neg(p \land q)) \\
&\Leftrightarrow \neg p \land \neg(\neg(p \land q)) && \textit{De Morgan's Law} \\
&\Leftrightarrow \neg p \land (p \land q) && \textit{Double Negation Law} \\
&\Leftrightarrow (\neg p \land p) \land q && \textit{Associative Law} \\
&\Leftrightarrow F \land q && \textit{Contradiction} \\
&\Leftrightarrow F && \textit{Domination Law and Commutative Law}
\end{aligned}$$

Question: Proof by Contradiction **[B ∧ (B ⇒ C)] ⇒ C**

Proof:

Suppose $[B \land (B \Rightarrow C)]$, to prove C

On contrary, assume $\neg C$

$\neg C \land [B \land (B \Rightarrow C)]$ must be true

$\Rightarrow \neg C \land [B \land (\neg B \lor C)]$

$\Rightarrow \neg C \land [(B \land \neg B) \lor (B \land C)]$

$\Rightarrow \neg C \land [f \lor (B \land C)]$

$\Rightarrow \neg C \land B \land C = \neg C \land C \land B = f \land B = f$

$\Rightarrow$ False, Contradiction $\Rightarrow C$

Question: Prove that ¬ B → ¬ (A ∧ (A → B)) by

a) Logical Equivalence **b) Rules of Inference** **c) Contradiction**

Answer

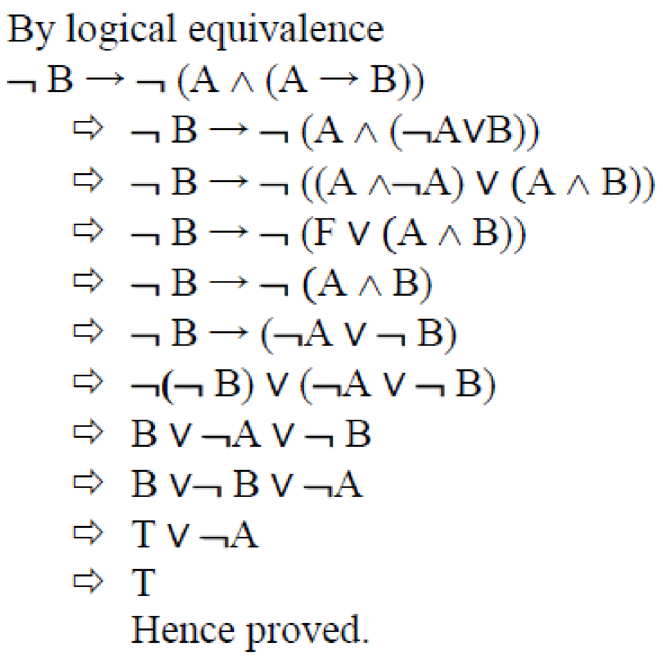
(a) Prove by Logical Equivalence ¬ B → ¬ (A ∧ (A → B)) (Solution-1)

¬ B → ¬ (A ∧ (A → B))

	Step	Description
⇒	¬ B → ¬ (A ∧ (¬ A ∨ B))	Implication Equivalence
⇒	¬ B → ¬ ((A ∧ ¬ A) ∨ (A ∧ B))	Distributive Law
⇒	¬ B → ¬ (false ∨ (A ∧ B))	Contradiction
⇒	¬ B → ¬ (A ∧ B)	Tautology / False is left one of ∨
⇒	¬ ¬ B ∨ ¬ (A ∧ B)	Implication Equivalence
⇒	B ∨ ¬ (A ∧ B)	Double Negation Law
⇒	B ∨ ¬ A ∨ ¬ B	De Morgan's Laws
⇒	B ∨ ¬ B ∨ ¬ A	Commutative Law
⇒	True ∨ ¬ A	Tautology
⇒	True	Domination

Therefore, proved that ¬ B → ¬ (A ∧ (A → B))

(Solution-2 By Instructor)



By logical equivalence

¬ B → ¬ (A ∧ (A → B))

⇨ ¬ B → ¬ (A ∧ (¬A∨B))

⇨ ¬ B → ¬ ((A ∧¬A) ∨ (A ∧ B))

⇨ ¬ B → ¬ (F ∨ (A ∧ B))

⇨ ¬ B → ¬ (A ∧ B)

⇨ ¬ B → (¬A ∨ ¬ B)

⇨ ¬(¬ B) ∨ (¬A ∨ ¬ B)

⇨ B ∨ ¬A ∨ ¬ B

⇨ B ∨¬ B ∨ ¬A

⇨ T ∨ ¬A

⇨ T

Hence proved.

(b) Prove by Rules of Inference $\neg B \rightarrow \neg (A \wedge (A \rightarrow B))$ **(Solution-1)**

A rule of inference is a general pattern that allows us to draw some new conclusion from a set of given statements. If we know A then we can conclude B.

In order to prove the given statement we will apply contraposition;

$\neg (\neg (A \wedge (A \rightarrow B))) \rightarrow \neg (\neg B)$

$\Rightarrow$ $(A \wedge (A \rightarrow B)) \rightarrow B$

Modus ponens: if $(A \wedge (A \rightarrow B))$ then B

Proof is as follows

Consider that $(A \wedge (A \rightarrow B))$ is true

$\Rightarrow$ A is true

$\Rightarrow$ $A \rightarrow B$ is True

$\Rightarrow$ B is True

Thus it is proved with contraposition that $(A \wedge (A \rightarrow B)) \rightarrow B$

Therefore,

$\neg B \rightarrow \neg (A \wedge (A \rightarrow B))$ is true

(Solution-2 By Instructor)

By rules of inference

Taking contraposition

$(A \wedge (A \rightarrow B)) \rightarrow B$

If A is true and $A\rightarrow B$ is true then B is true

Hence proved.

(c) Prove by Contradiction ¬ B → ¬ (A ∧ (A → B))

(Solution-1)

We will assume ¬ B to prove ¬ (A ∧ (A → B))

On contrary, suppose the statement (A ∧ (A → B)) is true;

	Step	**Description**
⇒	(A ∧ (A → B)) ∧ ¬ B	
⇒	(A ∧ (¬A ∨ B)) ∧ ¬ B	Implication Equivalence
⇒	((A ∧ ¬A) ∨ (A ∧ B)) ∧ ¬ B	Distributive Law
⇒	(false ∨ (A ∧ B)) ∧ ¬ B	Tautology
⇒	(A ∧ B) ∧ ¬ B	Contradiction
⇒	A ∧ B ∧ ¬ B	Distributive Law
⇒	A ∧ False	Contradiction
⇒	False	

Therefore proved that ¬ (A ∧ (A → B)) is true by contradiction

(Solution-2 by Instructor)

Suppose ¬ (A ∧ (A → B)) is false then (A ∧ (A → B)) is true

⇨ It means A is true and A → B is true

⇨ if A is true and A → B is true then B must also be true

⇨ but we have ¬ B true, which is contradiction

⇨ hence proved.

Question: Prove ~C → ~ (B ∧ (B → C))

1. By Logical Equivalences

2. Truth Table

3. Rules of inference

4. Contradiction

Solution:

1- By Logical Equivalence

$$\sim C \rightarrow \sim \left(B \wedge \left(B \rightarrow C\right)\right)$$

$\Rightarrow$ $\sim C \rightarrow \sim \left(B \wedge (\sim B \vee C)\right)$

$\Rightarrow$ $\sim C \rightarrow \sim \left(\left(B \wedge \sim B\right) \vee \left(B \wedge C\right)\right)$

$\Rightarrow$ $\sim C \rightarrow \sim \left(f \vee \left(B \wedge C\right)\right)$

$\Rightarrow$ $\sim C \rightarrow \sim \left(B \wedge C\right)$

$\Rightarrow$ $C \vee \sim \left(B \wedge C\right)$

$\Rightarrow$ $C \vee \sim B \vee \sim C$

$\Rightarrow$ $C \vee \sim C \vee \sim B$

$\Rightarrow$ $t \vee \sim B$

$\Rightarrow$ True

Hence proved that

$$\sim C \rightarrow \sim \left(B \wedge \left(B \rightarrow C\right)\right)$$

2- By Truth Table

B	C	~C	B→C	$B \wedge (B \to C)$	$\sim(B \wedge (B \to C))$	$\sim C \to \sim(B \wedge (B \to C))$
T	T	F	T	T	F	**T**
T	F	T	F	F	T	**T**
F	T	F	T	F	T	**T**
F	F	T	T	F	T	**T**

3. Rules of Inference

To prove $\sim C \to \sim(B \wedge (B \to C))$, we take its Contraposition as:

$$\sim(\sim(B \wedge (B \to C))) \ \to \ \sim(\sim C)$$

$$\Rightarrow \quad (B \wedge (B \to C)) \ \to \ C$$

Modus ponens

If $(B \wedge (B \to C))$ then C

Proof:

Suppose $(B \wedge (B \to C))$ is true then

$\Rightarrow \quad B$ is true and $B \to C$ is true.

$\Rightarrow \quad C$ is true.

Hence $(B \wedge (B \to C)) \ \to \ C$ is proved. Taking contraposition:

$\sim C \ \to \ \sim(B \wedge (B \to C))$ is also true.

4. Contradiction

Suppose $\sim C$, to prove $\sim\left(B \wedge (B \to C)\right)$

On contrary, assume $\left(B \wedge (B \to C)\right)$ is true

$\Rightarrow \quad \left(B \wedge (B \to C)\right) \wedge \sim C$

$\Rightarrow \quad \left(B \wedge (\sim B \vee C)\right) \wedge \sim C$

$\Rightarrow \quad ((B \wedge \sim B) \vee (B \wedge C)) \wedge \sim C$

$\Rightarrow \quad (f \vee (B \wedge C)) \wedge \sim C$

$\Rightarrow \quad (B \wedge C) \wedge \sim C$

$\Rightarrow \quad B \wedge C \wedge \sim C$

$\Rightarrow \quad B \wedge f$

$\Rightarrow \quad f$

$\Rightarrow$ False, Contradiction

Hence $\sim\left(B \wedge (B \to C)\right)$ is true.

Question: Show that (p ∨ q) ∧ (¬ p ∨ r) → (q ∨ r) is a Tautology?

Solution:

p	q	r	p ∨ q	¬ p	(¬ p ∨ r)	(p ∨ q) ∧ (¬ p ∨ r)	(q ∨ r)	(p ∨ q) ∧ (¬ p ∨ r) → (q ∨ r)
T	T	T	T	F	T	T	T	T
T	T	F	T	F	F	F	T	T
T	F	T	T	F	T	T	T	T
T	F	F	T	F	F	F	F	T
F	T	T	T	T	T	T	T	T
F	T	F	T	T	T	T	T	T
F	F	T	F	T	T	F	T	T
F	F	F	F	T	T	F	F	T

Since (p ∨ q) ∧ (¬ p ∨ r) → (q ∨ r) is always T, it is a tautology.

Question: Show that [(p → q) ∧ (q → r)] → (p → r) is a tautology?

Solution:

p	q	r	$p \to q$	$q \to r$	$(p \to q) \land (q \to r)$	$p \to r$	$[(p \to q) \land (q \to r)] \to (p \to r)$
T	T	T	T	T	T	T	T
T	T	F	T	F	F	F	T
T	F	T	F	T	F	T	T
T	F	F	F	T	F	F	T
F	T	T	T	T	T	T	T
F	T	F	T	F	F	T	T
F	F	T	T	T	T	T	T
F	F	F	T	T	T	T	T

Since $[(p \to q) \land (q \to r)] \to (p \to r)$ is always T, it is a tautology.

Question: Show $[(p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)] \rightarrow r$ is a Tautology?

Solution:

p	q	r	$p \vee q$	$p \rightarrow r$	$q \rightarrow r$	$(p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)$	$[(p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)] \rightarrow r$
T	T	T	T	T	T	T	T
T	T	F	T	F	F	F	T
T	F	T	T	T	T	T	T
T	F	F	T	F	T	F	T
F	T	T	T	T	T	T	T
F	T	F	T	T	F	F	T
F	F	T	F	T	T	F	T
F	F	F	F	T	T	F	T

Since $[(p \vee q) \wedge (p \rightarrow r) \wedge (q \rightarrow r)] \rightarrow r$ is always T, it is a tautology.

Question: Show $[\neg p \wedge (p \rightarrow q)] \rightarrow \neg q$ is a Not Tautology?

Solution:

p	q	$p \rightarrow q$	$\neg p \wedge (p \rightarrow q)$	$[\neg p \wedge (p \rightarrow q)] \rightarrow \neg q$
T	T	T	F	T
T	F	F	F	T
F	T	T	T	F
F	F	T	T	T

So $[\neg p \wedge (p \rightarrow q)] \rightarrow \neg q$ is not a tautology.

Question: Show that (p ∨ q) ∧ (¬p ∨ r) → (q ∨ r) is a Tautology?

Solution:

$(p \vee q) \wedge (\neg p \vee r) \to (q \vee r)$	
$\equiv \neg[(p \vee q) \wedge (\neg p \vee r)] \vee (q \vee r)$	by implication law
$\equiv [\neg(p \vee q) \vee \neg(\neg p \vee r)] \vee (q \vee r)$	by de Morgan's law
$\equiv [(\neg p \wedge \neg q) \vee (p \wedge \neg r)] \vee (q \vee r)$	by de Morgan's law
$\equiv [(\neg p \wedge \neg q) \vee q] \vee [(p \wedge \neg r) \vee r]$	by commutative and associative laws
$\equiv [(\neg p \vee q) \wedge (\neg q \vee q)] \vee [(p \vee r) \wedge (\neg r \vee r)]$	by distributive laws
$\equiv (\neg p \vee q) \vee (p \vee r)$	by negation and identity laws
$\equiv (\neg p \vee p) \vee (q \vee r)$	by communicative and associative laws
$\equiv T$	by negation and domination laws

Question: Prove that each of the following pairs of propositional formulae are equivalent using propositional equivalences.

(a) p ↔ q (p ∧ q) ∨ (¬p ∧ ¬q)

$p \leftrightarrow q$	$\equiv$	$(p \to q) \wedge (q \to p)$	[iff is two implications]
	$\equiv$	$(\neg p \vee q) \wedge (\neg q \vee p)$	[Law of Implication]
	$\equiv$	$((\neg p \vee q) \wedge \neg q) \vee ((\neg p \vee q) \wedge p)$	[Distributivity]
	$\equiv$	$(\neg p \wedge \neg q) \vee (q \wedge \neg q) \vee ((p \wedge \neg p) \vee (p \wedge q))$	[Distributivity]
	$\equiv$	$(\neg p \wedge \neg q) \vee (p \wedge q))$	[Negation]
	$\equiv$	$(p \wedge q) \vee (\neg p \wedge \neg q)$	[Commutativity]

(b) ¬p → (q → r) q → (p ∨ r)

$\neg p \to (q \to r)$	$\equiv$	$p \vee (\neg q \vee r)$	[Law of Implication]
	$\equiv$	$(p \vee \neg q) \vee r$	[Associativity]
	$\equiv$	$(\neg q \vee p) \vee r$	[Commutativity]
	$\equiv$	$\neg q \vee (p \vee r)$	[Associativity]
	$\equiv$	$q \to (p \vee r)$	[Law of Implication]

Question: Prove that each of the following propositional formulae are tautologies by showing they are equivalent to T.

(a). ((p → q) ∧ (q → r)) → (p → r)

$$
\begin{array}{lcll}
((p \to q) \wedge (q \to r)) \to (p \to r) & \equiv & \neg((\neg p \vee q) \wedge (\neg q \vee r)) \vee (\neg p \vee r) & \text{[Law Impl.]} \\
 & \equiv & ((p \wedge \neg q) \vee (q \wedge \neg r)) \vee (\neg p \vee r) & \text{[DeMorgan]} \\
 & \equiv & (((p \wedge \neg q) \vee (\neg p \vee r) \vee (q \wedge \neg r) & \text{[Associativity]} \\
 & \equiv & (((\neg p \vee r) \vee p) \wedge ((\neg p \vee r) \wedge \neg q)) \vee (q \wedge \neg r) & \text{[Distributivity]} \\
 & \equiv & (((p \vee \neg p) \vee r) \wedge ((\neg p \vee r) \vee \neg q)) \wedge (q \vee \neg r) & \text{[Assoc., Commut.]} \\
 & \equiv & ((\text{T} \vee r) \wedge ((\neg p \vee r) \vee \neg q)) \vee (q \wedge \neg r) & \text{[Negation]} \\
 & \equiv & (\text{T} \wedge ((\neg p \vee r) \vee \neg q)) \vee (q \wedge \neg r) & \text{[Domination]} \\
 & \equiv & ((\neg p \vee r) \vee \neg q) \vee (q \wedge \neg r) & \text{[Identity]} \\
 & \equiv & (q \vee ((\neg p \vee r) \vee \neg q) \wedge \neg r \vee ((\neg p \vee r) \vee \neg q)) & \text{[Distributivity]} \\
 & \equiv & ((q \vee \neg q) \vee (\neg p \vee r)) \wedge (\neg r \vee r \vee \neg p \vee \neg q) & \text{[Assoc., Comm.]} \\
 & \equiv & (\text{T} \vee (\neg p \vee r)) \wedge (\text{T} \vee \neg p \vee \neg q) & \text{[Negation]} \\
 & \equiv & \text{T} \wedge \text{T} & \text{[Domination]} \\
 & \equiv & \text{T} & \text{[Identity]}
\end{array}
$$

(b). (p ∧ q) ∨ (p ∧ r) → (q ∨ r)

$$
\begin{array}{lcll}
(p \wedge q) \vee (p \wedge r) \to (q \vee r) & \equiv & (p \wedge (r \vee q)) \to (q \vee r) & \text{[Distrib.]} \\
 & \equiv & \neg(p \wedge (r \vee q)) \vee (q \vee r) & \text{[Law of Implication]} \\
 & \equiv & \neg(p \vee \neg(r \vee q)) \vee (q \vee r) & \text{[DeMorgan]} \\
 & \equiv & \neg p \vee (\neg(r \vee q) \vee (r \vee q)) & \text{[Assoc., Comm.]} \\
 & \equiv & \neg p \vee \text{T} & \text{[Negation]} \\
 & \equiv & \text{T} & \text{[Identity]}
\end{array}
$$

(c). (p ∧ q) ∨ (¬p ∧ q) ∨ ¬q

$$
\begin{array}{lcll}
(p \wedge q) \vee (\neg p \wedge q) \vee \neg q & \equiv & (q \wedge (p \vee \neg p)) \vee \neg q & \text{[Comm., Assoc., Distrib.]} \\
 & \equiv & (q \wedge \text{T}) \vee \neg q & \text{[Negation]} \\
 & \equiv & q \vee \neg q & \text{[Identity]} \\
 & \equiv & \text{T} & \text{[Negation]}
\end{array}
$$

Question: Construct Truth Tables for each of the following compound propositions.

(a) (p ∧ q) ∨ (p ∧ r)

Solution:

(a)

p	q	r	$(p \wedge q) \vee (p \wedge r)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	F

(b) (q ∧ p) ↔ (q ⊕ p)

(b)

p	q	$(q \wedge p) \leftrightarrow (q \oplus p)$
T	T	F
T	F	F
F	T	F
F	F	T

Question: Are the following compound propositions Tautologies?

(a) $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$

(b) $((p \wedge q) \wedge (q \wedge r)) \rightarrow (p \wedge r)$

(c) $((p \oplus q) \wedge (q \oplus r)) \rightarrow (p \oplus r)$

In other words are the logical operator's implication, conjunction and exclusive-or transitive?

Solution:

(a) $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$ is a tautology because

p	q	r	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	T

(b) $((p \wedge q) \wedge (q \wedge r)) \rightarrow (p \wedge r)$

(b) $((p \wedge q) \wedge (q \wedge r)) \rightarrow (p \wedge r)$ is also a tautology because

p	q	r	$((p \wedge q) \wedge (q \wedge r)) \rightarrow (p \wedge r)$
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	T
F	F	T	T
F	F	F	T

(c) ((p ⊕ q) ∧ (q ⊕ r)) → (p ⊕ r)

This is not a tautology as seen by the case where p and r are *True* and q is *False*, or the case where p and r are *False* and q is *True*.

Thus the operator → and ∧ are transitive but ⊕ is not transitive.

Question: Let p, q, r denote primitive propositions. Determine whether the statements (p ∨ q) → r and (p → r) ∧ (q → r) are Logically Equivalent.

Solution:

The truth table for the formula [(p ∨ q) → r] ↔ [(p → r) ∧ (q → r)]:

p	q	r	$[(p \vee q)$	$\rightarrow r]$	$\leftrightarrow$	$[(p \rightarrow r)$	$\wedge$	$(q \rightarrow r)]$
T	T	T	T	T	T	T	T	T
T	T	F	T	F	T	F	F	F
T	F	T	T	T	T	T	T	T
T	F	F	T	F	T	F	F	T
F	T	T	T	T	T	T	T	T
F	T	F	T	F	T	T	F	F
F	F	T	F	T	T	T	T	T
F	F	F	F	T	T	T	T	T
		Step:	1	2	6	3	5	4

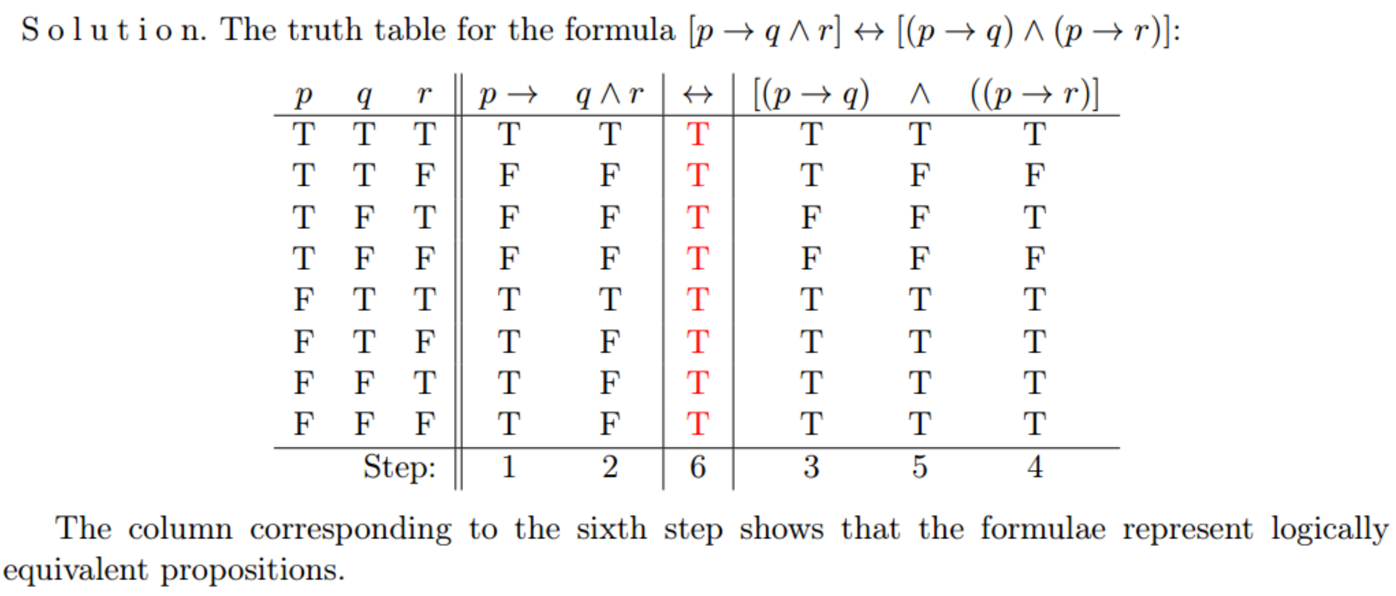
The column corresponding to the sixth step shows that the formulae represent logically equivalent propositions.

Alternative solution. Using the laws of logic we obtain:

$$\begin{aligned} (p \vee q) \rightarrow r &\equiv \neg(p \vee q) \vee r && \text{implication} \\ &\equiv (\neg p \wedge \neg q) \vee r && \text{De Morgan law} \\ &\equiv (\neg p \vee r) \wedge (\neg q \vee r) && \text{distributive law} \\ &\equiv (p \rightarrow r) \wedge (q \rightarrow r) && \text{implication} \end{aligned}$$

Question: Let p, q, r denote primitive propositions. Determine whether the statements $p \to q \land r$ and $(p \to q) \land (p \to r)$ are Logically Equivalent.

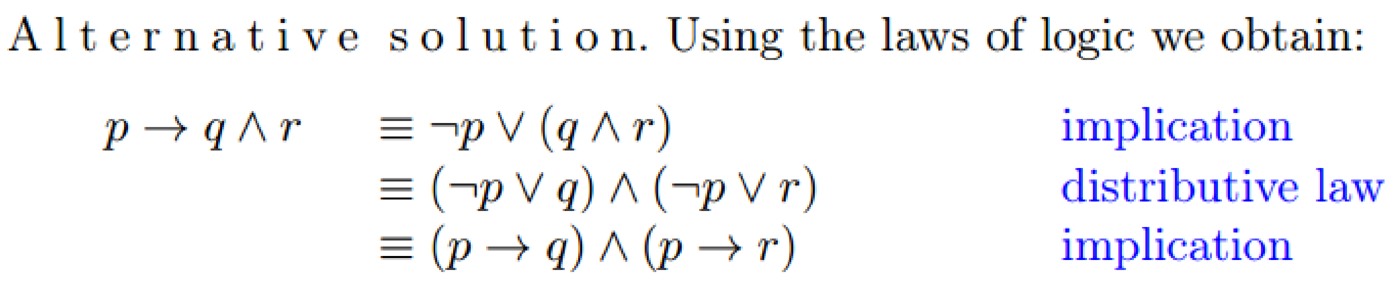
Solution:



S o l u t i o n. The truth table for the formula $[p \to q \land r] \leftrightarrow [(p \to q) \land (p \to r)]$:

p	q	r	$p \to$	$q \land r$	$\leftrightarrow$	$[(p \to q)$	$\land$	$((p \to r)]$
T	T	T	T	T	T	T	T	T
T	T	F	F	F	T	T	F	F
T	F	T	F	F	T	F	F	T
T	F	F	F	F	T	F	F	F
F	T	T	T	T	T	T	T	T
F	T	F	T	F	T	T	T	T
F	F	T	T	F	T	T	T	T
F	F	F	T	F	T	T	T	T
		Step:	1	2	6	3	5	4

The column corresponding to the sixth step shows that the formulae represent logically equivalent propositions.



A l t e r n a t i v e s o l u t i o n. Using the laws of logic we obtain:

$$\begin{aligned} p \to q \land r &\equiv \neg p \lor (q \land r) && \text{implication} \\ &\equiv (\neg p \lor q) \land (\neg p \lor r) && \text{distributive law} \\ &\equiv (p \to q) \land (p \to r) && \text{implication} \end{aligned}$$

Theorem

Let A and B are real numbers. A recurrence of the form

$a_k = Aa_{k-1} + Ba_{k-2}$ is satisfied by the

sequence $1, t, t^2, \ldots, t^n, \ldots t \neq 0$ where t is a non-zero real no, if and only if t satisfies the equation $t^2 - At - B = 0$

Proof:

Let us suppose that the sequence $1, t, t^2, \ldots, t^n, \ldots, where, t \neq 0$ satisfies the recurrence relation $a_k = Aa_{k-1} + Ba_{k-2}$. It means each form of sequence is equal to A times the previous form plus B times the form before it.

Hence $t^k = At^{k-1} + Bt^{k-2}$

since $t \neq 0 \Rightarrow t^{k-2} \neq 0$

Dividing both sides by t^{k-2}, we get

$t^2 - At - B = 0$

Conversely suppose that $t^2 - At - B = 0$

$\Rightarrow t^k = At^{k-1} + Bt^{k-2}$

Hence $1, t, t^2, t^3 \ldots, t^n \ldots$ satisfies the above recurrence relation.

Question: Using characteristic equation, find solution to a recurrence relation

$$a_k = a_{k-1} + 2a_{k-2} \quad \forall\, k \geq 2$$

Find all sequences that satisfy relation and have the form $1, t, t^2, \ldots, t^n, \ldots t \neq 0$

Solution: Above recurrence relation is satisfied by a sequence $1, t, t^2, \ldots, t^n, \ldots, t \neq 0$

if and only if, $t^2 - t - 2 = 0$

$\Rightarrow (t-2)(t+1) = 0 \qquad \Rightarrow t = 2, -1$

Hence

$2^0, 2, 2^2, 2^3, \ldots, 2^n, \ldots$ and $(-1)^0, (-1)^1, (-1)^2, (-1)^3, \ldots$

are both particular solutions for this recurrence relation.

Question: Solve the Recurrence / Solve First Order Recurrences

$$b_k = a.b_{k-1} \text{ if } b_0 = c$$

Solution:

$$b_k = a.b_{k-1}$$
$$b_k = a.(a.b_{k-2})$$
$$b_k = a^2.(a.b_{k-3})$$
$$\ldots$$
$$b_k = a^k.b_{k-k} = a^k.b_0$$
$$b_k = a^k.c$$

Now the explicit formula is $c.a^k$ which is a geometric sequence.
Hence first order recurrence is nothing but G.S.

Question: Solve the Recurrence $b_k = 5b_{k-1}$ if $b_0 = 3$

Solution:

$$b_k = 5.b_{k-1}$$
$$b_k = 5.(5.\ b_{k-2}) = 5^2.\ b_{k-2}$$
$$b_k = 52.(5.\ b_{k-3}) = 5^3.\ b_{k-3}$$
$$\ldots$$
$$b_k = 5k.\ b_{k-k} = 5^k.\ b_0 = 5^k.3$$

Now the solution sequence becomes

$3.5^0, 3.5^1, 3.5^2, 3.5^3, 3.5^4, \ldots, 3.5^k, \ldots$

3, 15, 75, 375, . . .geometric sequence

Question: Suppose sequence, $b_0, b_1, b_2, \ldots$ satisfies recurrence relation

$$b_k = 4b_{k-1} - 4b_{k-2} \qquad \forall\, k \geq 2$$

With initial condition: $b_0 = 1$ and $b_1 = 3$

Then find explicit formula for $b_0, b_1, b_2, \ldots,$ using characteristic equation of the above recursion.

<u>**Solution:**</u> Characteristic equation is $t^2 - 4t + 4 = 0$

$(t-2)^2 = 0 \Rightarrow t = 2$, is repeated root

2^n and $n.2^n$ are sequences which satisfy the same recurrence relation, but do not satisfy the initial conditions

Suppose general solution: $b_n = C.2^n + D.n.2^n$

which satisfies the original recurrence , C and D are constants

Since $b_0 = 1,\quad b_1 = 3$

For $n = 0,\quad C + D \cdot 0 \cdot 2^0 = 1 \Rightarrow \quad C = 1$

For $n = 1,\quad b_1 = C \cdot 2^1 + D \cdot 1 \cdot 2^1$

$$\Rightarrow\ 2C + 2D = 3 \Rightarrow 2 \cdot 1 + 2D = 3 \Rightarrow\ D = \frac{3-2}{2} = \frac{1}{2}$$

$$\text{Hence, } b_n = 1 \cdot 2^n + \frac{1}{2} \cdot n \cdot 2^n = 2^n\left(1 + \frac{n}{2}\right)$$

$$= \left(1 + \frac{n}{2}\right)2^n \text{ is the required solution for repeated roots.}$$

Checking explicit formula:

$b_n = (1 + n/2).2^n$

First term	$b_0 = (1 + 0/2).2^0 = 1.1 = 1$
Second term	$b_1 = (1 + 1/2).2^1 = 3/2.2 = 3$
Third term	$b_2 = (1 + 2/2).2^2 = 2.4 = 8$
Fourth term	$b_3 = (1 + 3/2).2^3 = 5/2.8 = 20$, and so on

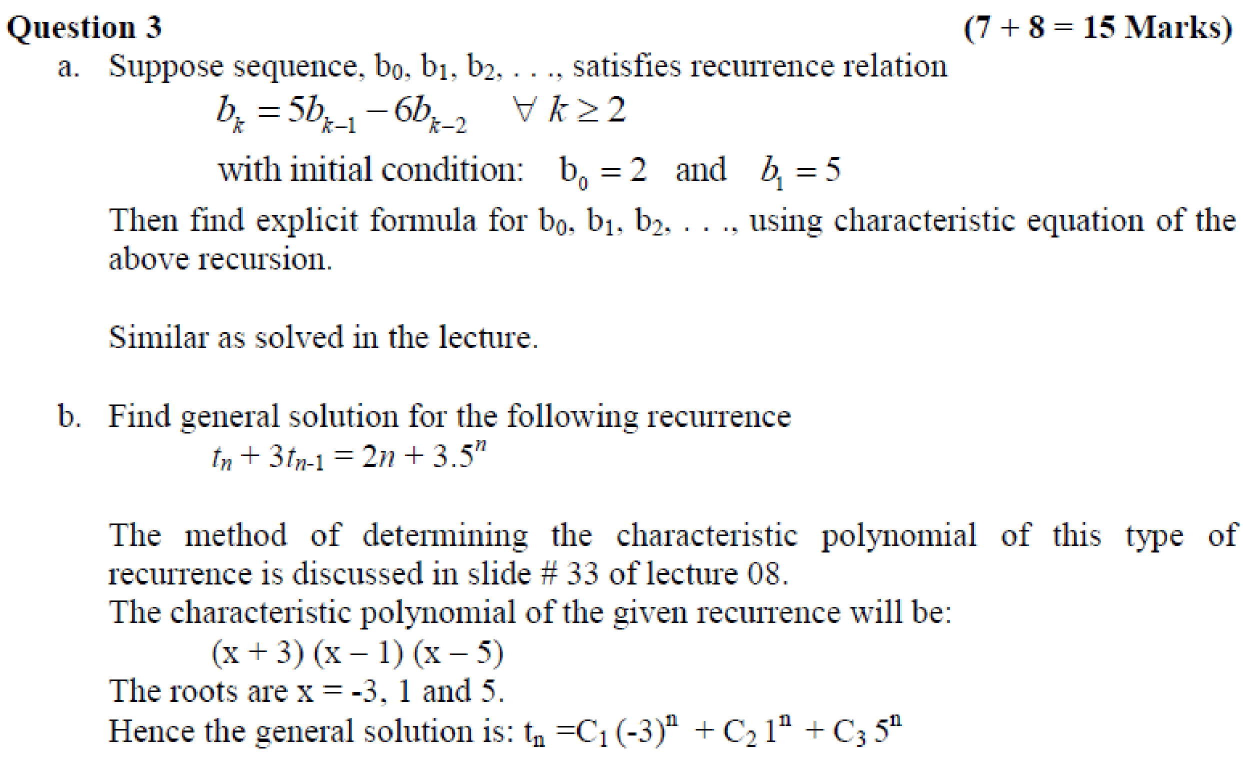
Question: Suppose sequence, b_0, b_1, b_2, . . . , satisfies Recurrence Relation

$$b_k = 5b_{k-1} - 6b_{k-2} \; \forall \; k \geq 2$$

With initial condition: $b_0 = 2$ and $b_1 = 5$

Then find explicit formula for b_0, b_1, b_2 , . . . , using characteristic equation of the above recursion.

Solution:



Question 3 **(7 + 8 = 15 Marks)**

a. Suppose sequence, b_0, b_1, b_2, . . ., satisfies recurrence relation

$$b_k = 5b_{k-1} - 6b_{k-2} \quad \forall\, k \geq 2$$

with initial condition: $b_0 = 2$ and $b_1 = 5$

Then find explicit formula for b_0, b_1, b_2, . . ., using characteristic equation of the above recursion.

Similar as solved in the lecture.

b. Find general solution for the following recurrence

$$t_n + 3t_{n-1} = 2n + 3.5^n$$

The method of determining the characteristic polynomial of this type of recurrence is discussed in slide # 33 of lecture 08.
The characteristic polynomial of the given recurrence will be:

$$(x + 3)\,(x - 1)\,(x - 5)$$

The roots are x = -3, 1 and 5.
Hence the general solution is: $t_n = C_1\,(-3)^n + C_2\,1^n + C_3\,5^n$

Question: Solve the recurrence $b_k = a.b_{k-1}$ if $b_0 = c$

Solution

$$b_k = a.b_{k-1}$$
$$b_k = a.(a.b_{k-2})$$
$$b_k = a^2.(a.b_{k-3})$$
$$\dots$$
$$b_k = a^k.b_{k-k} = a^k.b_0$$
$$b_k = a^k.c$$

Now the explicit formula is $c.a^k$ which is a geometric sequence. Hence first order recurrence is nothing but G.S.

Question: Solve the recurrence $b_k = 5b_{k-1}$ if $b_0 = 3$

Solution

$$b_k = 5.b_{k-1}$$
$$b_k = 5.(5.\, b_{k-2}) = 5^2.\, b_{k-2}$$
$$b_k = 52.(5.\, b_{k-3}) = 5^3.\, b_{k-3}$$
$$\dots$$
$$b_k = 5k.\, b_{k-k} = 5^k.\, b_0 = 5^k.3$$

Now the solution sequence becomes

$3.5^0, 3.5^1, 3.5^2, 3.5^3, 3.5^4, \dots, 3.5^k, \dots$

3, 15, 75, 375, . . .geometric sequence

Question: Consider the Recurrence

$$t_n = n \text{ if } n = 0, 1, 2$$

$$t_n = 7.t_{n-2} + 6.t_{n-3} \qquad \text{otherwise}$$

Find the general solution of the recurrence above.

<u>Solution:</u> First we rewrite the recurrence.

$$t_n = 7.t_{n-2} + 6.t_{n-3}$$

The characteristic equation becomes $x^3 - 7x - 6 = (x + 1)(x + 2)(x - 3)$

The roots are: $r_1 = -1$, $r_2 = -2$ and $r_3 = +3$

$$t_n = c_1 (-1)^n + c_2 (-2)^n + c_3 (3)^n$$

The initial conditions give

$c_1 + c_2 + c_3 = 0$ for $n = 0$

$-c_1 - 2c_2 + 3c_3 = 1$ for $n = 1$

$c_1 + 4c_2 + 9c_3 = 2$ for $n = 2$

Solving these equations, we obtain

$c_1 = -1/4$, $c_2 = 0$ and $c_3 = 1/4$

Therefore, $t_n = (-1/4)(-1)^n + (1/4).(3)^n$

Question: Consider the Recurrence

$t_n = n$ if $n = 0, 1, 2$

$t_n = 5t_{n-1} - 8t_{n-2} + 4t_{n-3}$ otherwise

Find general solution of the recurrence above.

Solution: First we rewrite the recurrence.

$$t_n - 5t_{n-1} + 8t_{n-2} - 4t_{n-3} = 0$$

The characteristic equation become.

$$x^3 – 5x^2 + 8x -4 = (x-1)(x-2)^2$$

The roots are: $r_1 = 1$ of multiplicity $m_1 = 1$ and $r_2 = 2$ of multiplicity $m_2 = 2$, and hence the general solution is $t_n = c_1 1^n + c_2 2^n + c_3 n 2^n$

The initial conditions give

$c_1 + c_2 = 0$ for $n = 0$

$c_1 + 2c_2 + 2c_3 = 1$ for $n = 1$

$c_1 + 4c_2 + 8c_3 = 2$ for $n = 2$

Solving these equations, we obtain

$c_1 = -2$, $c_2 = 2$ and $c_3 = -1/2$

Therefore,

$$t_n = c1\ 1^n + c_2 2^n + c_3 n 2^n$$
$$= -2 + 2.2^n – ½.n.2^n = 2^{n+1} – n.2^{n-1} - 2$$

Generalization: Non-homogeneous Recurrences

Question: Consider the Recurrence below. Find its solution

$$t_n - 2t_{n-1} = 3^n$$

Solution:

Compare the above recurrence with

$$a_0 t_n + a_1 t_{n-1} + \ldots + a_k t_{n-k} = b^n p(n)$$

Here: $b = 3$, $p(n) = 1$, a polynomial of degree 0.

Reducing to homogeneous case, we are given

$$t_n - 2t_{n-1} = 3^n \qquad (1)$$

Replace n by n - 1 and multiply by 3, we obtain

$$t_{n-1} - 2t_{n-2} = 3^{n-1}$$

$$3t_{n-1} - 6t_{n-2} = 3^n \qquad (2)$$

From (1) and (2)

$$t_n - 2t_{n-1} = 3^n \qquad (1)$$

$$+ 3t_{n-1} - 6t_{n-2} = 3^n \qquad (2)$$

Subtracting 2 from equation 1, we get

$$t_n - 5t_{n-1} + 6t_{n-2} = 0$$

The characteristic equation is $x^2 - 5x + 6 = 0$

Roots of this equation are: $x = 2$ and $x = 3$

And therefore general solution is $t_n = c_1 2^n + c_2 3^n$

It is not always true that an arbitrary choice of c_1 and c_2 produces a solution to the recurrence even when initial conditions are not taken into account.

Problem: Tower of Hanoi

- Tower of Hanoi is a mathematical game or puzzle. It consists of three towers, a number of disks of different sizes which can slide onto any tower. The puzzle starts with disks stacked in order of size on one tower, smallest at top, making a conical shape.
- Objective is to move entire stack to another tower, obeying following rules:
- Only one disk may be moved at a time.
- Each move consists of taking upper disk from one of towers and sliding it onto another tower
- You can put on top of other disks already present
- No disk may be placed on top of a smaller disk.
- If a recurrence is of the form $a_0 t_n + a_1 t_{n-1} + \ldots + a_k t_{n-k} = b_1^n p_1(n) + b_2^n p_2(n) + \ldots$
- Then the characteristics polynomial is $(a_0 x^k + a_1 x^{k-1} + \ldots + a_k)(x - b_1^{d_1+1})(x - b_2^{d_2+1})\ldots,$
- Which contains one factor for the left hand side
- And other factor corresponding to the each term on right hand side.
- Once the characteristics polynomial is obtained the recurrence can be solved as before.

Question: Find General Solution of the following Recurrence.

$$t_n - 2t_{n-1} = (n + 5)\, 3^n ; \qquad n \geq 1$$

Solution:

The manipulation needed to transform this into a homogeneous recurrence is slightly more complicated than with first example.

$t_n - 2t_{n-1} = (n + 5)\, 3^n$; $n \geq 1$ (1)

replace n by n-1, n-2, we get

$t_{n-1} - 2t_{n-2} = (n + 4)\, 3^{n-1}$; $n \geq 2$ (2)

$t_{n-2} - 2t_{n-3} = (n + 3)\, 3^{n-2}$; $n \geq 3$ (3)

Above equations can be written as

$t_n - 2t_{n-1} = 9(n + 5)\, 3^{n-2}$; $n \geq 1$ (4)

$t_{n-1} - 2t_{n-2} = 3(n + 4)\, 3^{n-2}$; $n \geq 2$ (5)

$t_{n-2} - 2t_{n-3} = (n + 3)\, 3^{n-2}$; $n \geq 3$ (6)

Our objective is to eliminate the right hand side of the above equations to make it homogenous.

Multiply (5) by -6, and (6) by 9 we get

$t_n - 2t_{n-1} = 9(n + 5)\, 3^{n-2}$

$- 6t_{n-1} + 12t_{n-2} = -18(n + 4)\, 3^{n-2}$

$+ 9t_{n-2} - 18t_{n-3} = 9(n + 3)\, 3^{n-2}$

After simplification, the above equations can be written as

$t_n - 2t_{n-1} = (9n + 45)\, 3^{n-2}$

$- 6t_{n-1} + 12t_{n-2} = (-18n – 72)\, 3^{n-2}$

$+ 9t_{n-2} - 18t_{n-3} = (9n + 27)\, 3^{n-2}$

Adding these equation, we get homogenous equation, which can be solved easily

$t_n - 8t_{n-1} + 21t_{n-2} - 18t_{n-3} = 0$

The characteristics equation of the above homogenous equation is:

$x^3 – 8x^2 +21x -18 = 0$

$(x-2)\, (x-3)^2 = 0$

and hence, $x = 2, 3, 3$

General solution is: $t_n = c_1\, 2^n + c_2\, 3^n + c_3\, n\, 3^n$

For n = 0, 1, 2

We can find values of c_1, c_2, c_3 and then

$t_n = (t_0 - 9)\, 2^n + (n + 3)3^{n+1}$

Non-Homogeneous Recurrences

Question: Consider the recurrence

$$t_n = 2t_{n-1} + n + 2^n \qquad \text{otherwise}$$

Solution: Compare the recurrence: $t_n - 2t_{n-1} = n + 2^n$ with

$$a_0 t_n + a_1 t_{n-1} + . . . + a_k t_{n-k} = b_1^n p_1(n) + b_2^n p_2(n) + \ldots$$

Here, $b_1 = 1$, $p_1(n) = n$, $b_2 = 2$, and $p_2(n) = 1$.

Degree of $p_1(n) = d_1 = 1$,

Degree of $p_2(n) = d_2 = 0$.

The characteristic polynomial: $(x-2)(x-1)^2(x-2)$

The roots are, $x = 1, 2$, both of multiplicity 2.

All solutions of recurrence therefore have form

$$t_n = c_1 1^n + c_2 n1^n + c_3 2^n + c_4 n 2^n$$

$$n + 2^n = (2c_2 - c_1) - c_2 n + c_4 2^n$$

For $n = 0, 1, 2, 3$ c_1, c_2, c_3 and c_4 can be solved and hence solution is

$$t_n = n.2^n + 2^{n+1} - n - 2$$

Substitution Method

Question: Solve the recurrence relation T(n) given below. [Otherwise 3T(n/4) + n]

$$T(n) = \begin{cases} 1 & if\ n = 1 \\ 3.T\left(\frac{n}{4}\right) + n & otherwise \end{cases}$$

Solution: (1) $T(n) = 3 \cdot T(\frac{n}{4}) + n$

replace the value of n by n/4 in (1)

(2) $T(\frac{n}{4}) = 3 \cdot T(\frac{n}{4^2}) + \frac{n}{4}$

(3) $T(\frac{n}{4^2}) = 3 \cdot T(\frac{n}{4^3}) + \frac{n}{4^2}$ and so on

(4) $T(\frac{n}{4^{k-1}}) = 3 \cdot T(\frac{n}{4^k}) + \frac{n}{4^{k-1}}$

Now substitute the value of $T(\frac{n}{4})$ from (2) to (1)

(5) $T(n) = 3\left[3T(\frac{n}{4^2}) + \frac{n}{4}\right] + n$

$$= 3^2 \cdot T(\frac{n}{4^2}) + (\frac{3}{4}) \cdot n + n$$

substitute the value of $T(\frac{n}{4^2})$ from (3) to (5) equation, we have

$$T(n) = 3^2\left[3 \cdot T(\frac{n}{4^3}) + \frac{n}{4^2}\right] + (\frac{3}{4})n + n$$

After continuing this process

$$T(n) = 3^k \cdot T(\frac{n}{4^k}) + (\frac{3}{4})^{k-1} n + (\frac{3}{4})^{k-2} + \cdots + (\frac{3}{4})n + (\frac{3}{4})^0 n$$

Let us suppose that n can be expressed as $n = 4^k$

$$T(n) = 3^k \cdot T(\frac{n}{n}) + n\left[1 + (\frac{3}{4}) + (\frac{3}{4})^2 + \ldots. + (\frac{3}{4})^{k-1}\right]$$

$$T(n) = 3^k \cdot T(1) + n\left[1 \cdot (\frac{1-(\frac{3}{4})^k}{1-\frac{3}{4}})\right] \qquad \because 1 + x + x^2 + \ldots + x^{k-1} = 1 \cdot (\frac{1-x^k}{1-x})$$

$$T(n) = 3^k \cdot 1 + 4n(1 - \frac{3^k}{4^k}) \qquad \because T(1) = 1$$

$$T(n) = 3^k + 4n \cdot (1 - \frac{3^k}{4^k}) = 3^k + 4n \cdot (1 - \frac{3^k}{n})$$

$$= 3^k + 4n(\frac{n-3^k}{n})$$

$$= 3^k + 4(n - 3^k) = 1 \cdot 3^k + 4n - 4 \cdot 3^k$$

$$= 4 \cdot n - 3 \cdot 3^k = 4n - 3 \cdot 3^{\log_4^n} \qquad \because 4^k = n, \quad k = \log_4^n$$

$$T(n) = 4n - 3 \cdot 3^{\log_4^n}$$

$$T(n) = 4n - 3 \cdot n^{\log_4^3}$$

$$T(n) = 4n - 3 \cdot n^{\alpha} \qquad \because 0 \le \alpha \le 1$$

$$\Rightarrow \; T(n) = 4n - 3 \cdot n^{\alpha}$$

$$Hence \quad T(n) \in \theta(n) \qquad let \quad \alpha = \log_4^3$$

Question: Consider the recurrence

$$t_n = n \qquad \text{if } n = 0, 1, 2$$

$$t_n = 6.t_{n-1} - 11.t_{n-2} + 6.t_{n-3} \qquad \text{otherwise}$$

Find the general solution of the recurrence above.

Solution:

Rewriting the recurrence,

$t_n - 6.t_{n-1} + 11.t_{n-2} - 6.t_{n-3} = 0$

The characteristic equation becomes

$x^3 – 6x^2 + 11x – 6 = 0$

Testing whether 1 or -1 is a root of the equation or not, we find that 1 is a root of the equation, hence

$(x-1)(x^2 – 5x + 6) = 0$

$\Rightarrow (x-1)(x-2)(x-3) = 0$

So the root are $r_1 = 1$, $r_2 = 2$ and $r_3 = 3$

$t_n = c_1 (1)^n + c_2 (2)^n + c_3 (3)^n$

the initial conditions give

$c_1 + c_2 + c_3 = 0$ for $n = 0$

$c_1 + 2c_2 + 3c_3 = 1$ for $n = 1$

$c_1 + 4c_2 + 9c_3 = 2$ for $n = 2$

Solving equations we obtain,

$c_1 = -3/2$, $c_2 = 2$, $c_3 = -1/2$

Hence, $t_n = (-3/2).(1)^n + 2.(2)^n + (-1/2).(3)^n$ is the general solution of the recurrence.

Question: Consider the recurrence given below. Find its general solution.

$$t_n - 4t_{n-1} = (n + 3)5^n$$

Solution:

Finding general solution of the following recurrence

$\mathbf{t_n - 4t_{n-1} = (n + 3)\, 5^n}$ **n >= 1** **(1)**

Replacing n with n-1

$t_{n-1} - 4t_{n-2} = (n + 2)\, 5^{n-1}$ $n >= 2$ (2)

Replacing n with n-2 we get

$t_{n-2} - 4t_{n-3} = (n + 1)\, 5^{n-2}$ $n >= 3$ (3)

Above equations can be written as follows

$t_n - 4t_{n-1} = 25(n + 3)\, 5^{n-2}$ $n >= 1$ (4) [from eq.1 5^2, n-1]

$t_{n-1} - 4t_{n-2} = 5(n + 2)\, 5^{n-2}$ $n >= 2$ (5) [from eq.2, n-1-1]

$t_{n-2} - 4t_{n-3} = (n + 1)\, 5^{n-2}$ $n >= 3$ (6) [from eq.3]

Our objective is to eliminate the right hand side of the above equations to make it homogenous.

Multiply equation (5) by -10 and equation (6) by 25 we get

$t_n - 4t_{n-1} = 25(n + 3)\, 5^{n-2}$

$-10t_{n-1} + 40t_{n-2} = -50(n + 2)\, 5^{n-2}$

$25t_{n-2} - 100t_{n-3} = 25(n + 1)\, 5^{n-2}$

Adding the above equations we get

$t_n - 14t_{n-1} + 65t_{n-2} - 100t_{n-3} = 0$

The characteristics equation of the above homogenous equation is

$x^3 - 14x^2 + 65x - 100 = 0$

$\Rightarrow (x - 4)\,(x^2 - 10x + 25) = 0$

$\Rightarrow (x - 4)\,(x - 5)^2 = 0$

Hence x = 4, 5, 5

The general solution is

$t_n = c_1.4^n + c_2.5^n + c_3.n5^n$

Question: Find general solution of the following recurrence.

$$t_n - 4t_{n-1} = (n + 5)\, 3^n \,,\ n >= 1$$

Solution:

Finding general solution of the following recurrence

$t_n - 4t_{n-1} = (n + 5)\, 3^n \quad n >= 1 \quad (1)$

Replacing n with n-1

$t_{n-1} - 4t_{n-1-1} = (n-1 + 5)\, 3^{n-1}$

$t_{n-1} - 4t_{n-2} = (n + 4)\, 3^{n-1} \quad n >= 2 \quad (2)$

Replacing n with n-2 we get

$t_{n-2} - 4t_{n-2-1} = (n-2 + 5)\, 3^{n-2}$

$t_{n-2} - 4t_{n-3} = (n + 3)\, 3^{n-2} \quad n >= 3 \quad (3)$

Above equations can be written as follows

$t_n - 4t_{n-1} = 9(n + 5)\, 3^{n-2} \quad n >= 1 \quad$ (4) [from eq.1 3^2, n-1]

$t_{n-1} - 4t_{n-2} = 3(n + 4)\, 3^{n-2} \quad n >= 2 \quad$ (5) [from eq.2, n-1-1]

$t_{n-2} - 4t_{n-3} = (n + 3)\, 3^{n-2} \quad n >= 3 \quad$ (6) [from eq.3]

Our objective is to eliminate the right hand side of the above equations to make it homogenous. So, multiply equation (6) by -12 we get

$-12t_{n-2} + 48t_{n-3} = -12(n + 3)\, 3^{n-2} \quad (7)$

Adding equation (4), (5) and (7)

$t_n - 3t_{n-1} - 16t_{n-2} + 48t_{n-3} = 0$

The characteristics equation of the above homogenous equation is

$x^3 – 3x^2 - 16x + 48 = 0$

$\Rightarrow (x-3)(x+4)(x-4)$

Hence x = 3, -4, 4

The general solution is

$t_n = c_1.3^n – c_2.4^n + c_3.n4^n$

Substitution Method

Question: Solve the recurrence relation T(n) given below using substitution method?

[4T(n/5) + n]

$$T(n) = \begin{cases} 1 & if\ n = 1 \\ 4.T\left(\frac{n}{5}\right) + n & otherwise \end{cases}$$

Solution:

$$T(n) = \begin{cases} 1 & if\ n = 1 \\ 4.T\left(\frac{n}{5}\right) + n & otherwise \end{cases}$$

By substitution method

$$T(n) = 4.T\left(\frac{n}{5}\right) + n \qquad (1)$$

Replace the value of n by $\frac{n}{5}$ $in\ (1)$

$$T\left(\frac{n}{5}\right) = 4.T\left(\frac{n}{5^2}\right) + \frac{n}{5} \qquad (2)$$

$$T\left(\frac{n}{5^2}\right) = 4.T\left(\frac{n}{5^3}\right) + \frac{n}{5^2} \qquad (3)$$

....

$$T\left(\frac{n}{5^{k-1}}\right) = 4.T\left(\frac{n}{5^k}\right) + \frac{n}{5^{k-1}} \qquad (4)$$

Now back substitute the value of $T\left(\frac{n}{5}\right)$ from (2) to (1)

$$T(n) = 4\left[4.T\left(\frac{n}{5^2}\right) + \frac{n}{5}\right] + n$$

$$= 4^2.T\left(\frac{n}{5^2}\right) + \left(\frac{4}{5}\right)n + n \qquad (5)$$

Substitute the value of $T\left(\frac{n}{5^2}\right)$ $from\ (3)\ to\ (5)$

$$T(n) = 4^2\left[4.T\left(\frac{n}{5^3}\right) + \frac{n}{5^2}\right] + \left(\frac{4}{5}\right)n + n$$

$$= 4^3.T\left(\frac{n}{5^3}\right) + \frac{4^2}{5^2}n + \frac{4}{5}n + n$$

. . . .

$$T(n) = 4^k.T\left(\frac{n}{5^k}\right) + \left(\frac{4}{5}\right)^{k-1} n + \left(\frac{4}{5}\right)^{k-2} n + \ldots + \left(\frac{4}{5}\right) n + \left(\frac{4}{5}\right)^0 n$$

Let us suppose that n can be expressed as n = 5^k

$$T(n) = 4^k.T\left(\frac{n}{n}\right) + n\left[1 + \frac{4}{5} + \left(\frac{4}{5}\right)^2 + \left(\frac{4}{5}\right)^3 + \ldots + \left(\frac{4}{5}\right)^{k-1}\right]$$

$$T(n) = 4^k \cdot T(1) + n\left[1 \cdot \left(\frac{1-(\frac{4}{5})^k}{1-\frac{4}{5}}\right)\right] \qquad \because 1 + x + x^2 + \ldots + x^{k-1} = 1 \cdot \left(\frac{1-x^k}{1-x}\right)$$

$$T(n) = 4^k \cdot 1 + 5n\left(1 - \frac{4^k}{5^k}\right) \qquad \because T(1) = 1$$

$$T(n) = 4^k + 5n\left(1 - \frac{4^k}{n}\right)$$

$$T(n) = 4^k + 5\left(n - 4^k\right)$$

$$T(n) = 5n - 4.(4)^k$$

$$T(n) = 5n - 4.4^{\log_5 n} \qquad \because 5^k = n \ \ and \ \ k = \log_5 n$$

$$T(n) = 5n - 4.n^{\log_5 4}$$

$$T(n) = 5n - 4n^{\alpha} \qquad Let\ 0 \le \alpha \le 1$$

$$Hence \quad T(n) \in \theta(n)$$

Recursive Tree Method

Question: Solve the recurrence relation given below using recursive tree method.

[3T(n/2) + n^2]

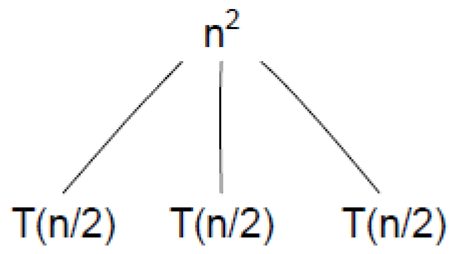
$$T(n) = \begin{cases} 1 & if\ n = 1 \\ 3.T\left(\frac{n}{2}\right) + n^2, & otherwise \end{cases}$$

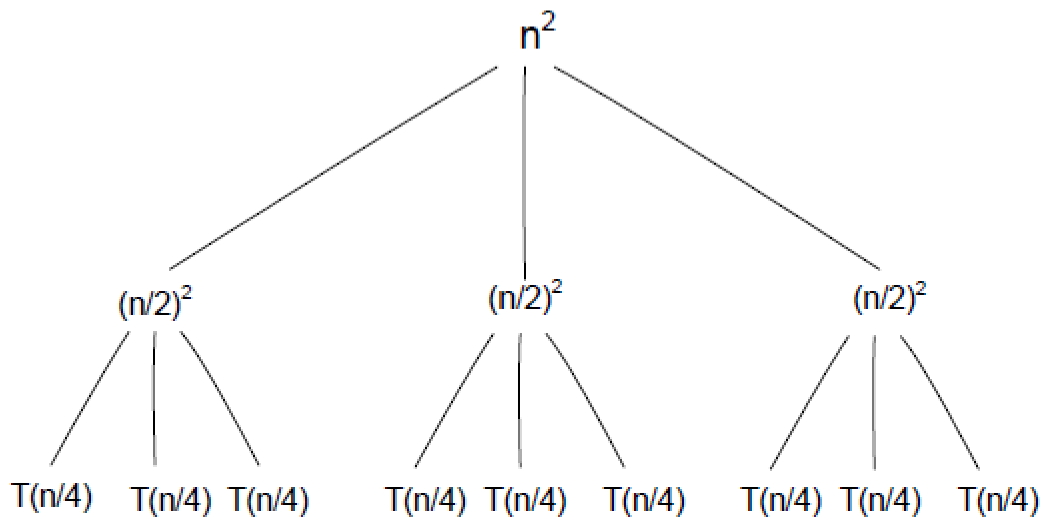
Solution

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3 \cdot T(\frac{n}{2}) + n^2 & \text{otherwise} \end{cases}$$

By Recursive Tree Method.

$T(n) = 3.T(n/2)+n^2$





……………………………… and so on

The subproblem size at depth i is $n/2^i$. The subproblem size hits n = 1 when $n/2^i = 1$ or equivalently when $i = \log_2 n$. Thus, the tree has total $\log_2 n + 1$ levels.
Each level has three times more nodes than the level above it, therefore, at level i, there will 3^i no. of nodes, and hence the total cost at level i will be $3^i . (n/2^i)^2 = (3/4)^i . n^2$. The last level at depth $\log_2 n$, there will be $3^{\log_2 n}$ or equivalently $n^{\log_2 3}$ nodes, each with cost T(1). Hence, total cost at last level will be $n^{\log_2 3}$ T(1) = $\Theta(n^{\log_2 3})$.

$$T(n) = \Theta(n^{\log_2^3}) + \left[(\frac{3}{4})^0 + (\frac{3}{4})^1 + \cdots + (\frac{3}{4})^{\log_2 n - 1} \right] n^2$$

$$\leq \Theta(n^{\log_2^3}) + \left[(\frac{3}{4})^0 + (\frac{3}{4})^1 + \cdots \right] n^2$$

$$= \Theta(n^{\log_2^3}) + (\frac{1}{1 - (\frac{3}{4})}) cn^2 = \Theta(n^{\log_2^3}) + 4n^2$$

Hence, $T(n) = O(n^2)$

Question: Use Mathematical Induction to prove that sum of first m positive integers is m(m+1)/2?

OR

Use Mathematical Induction to prove

$$\frac{m(m+1)}{2}$$

Proof:

Let P(m) denote the proposition that $1 + 2 + 3 + . . . + m = \frac{m(m+1)}{2}$

Basis step: P(1) is true, since 1 = (1(1+1)) / 2

Inductive step: Let P(k) is true for a positive integer k, i.e. $1 + 2 + . . . + k = \frac{k(k+1)}{2}$

Note that $1 + 2 . . . + k + k+1 = \frac{k(k+1)}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$

$\therefore$ P(k+1) true, by induction, P(n) is true for all n $\in Z^+$.

Question: Show that in any base $b \geq 2$, the sum of any three single-digit numbers is no more than two digits long.

Solution [1]

A single digit number is at most $b-1$, therefore the sum of any three such numbers is at most $3(b-1)=3b-3$.

On the other hand, a two-digit number can be as large as b^2-1.

It is enough to show that $b^2-1 \geq 3b-3$.

Indeed, $b^2 - 1 - 3b + 3 = (b-1) \cdot (b-2)$, which is ≥ 0 for $b \geq 2$.

Solution [2]: By Induction Method

Basis step:

Maximum possible digit in base 2 is 1, adding three 1_2 gives

$1_2 + 1_2 + 1_2 = 11_2$

We know that greatest digit in base b is (b-1), adding three such digits in base b gives

$(b-1)_b + (b-1)_b + (b-1)_b = (2\ (b-3))_b$ (1)

where $(2\ (b-3))_b$ is a two digit number in base b

And 2 is the most significant digit and (b-3) is the least significant digit.

We have to (1) by induction

Basis step:

$b = 3$,

$2_3 + 2_3 + 2_3 = (20)_3$

Inductive step

let (1) is true for $b = k$ i.e.

$(k-1)_k + (k-1)_k + (k-1)_k = (2\ (k-3))_k$

now we have to prove it for $b = k+1$

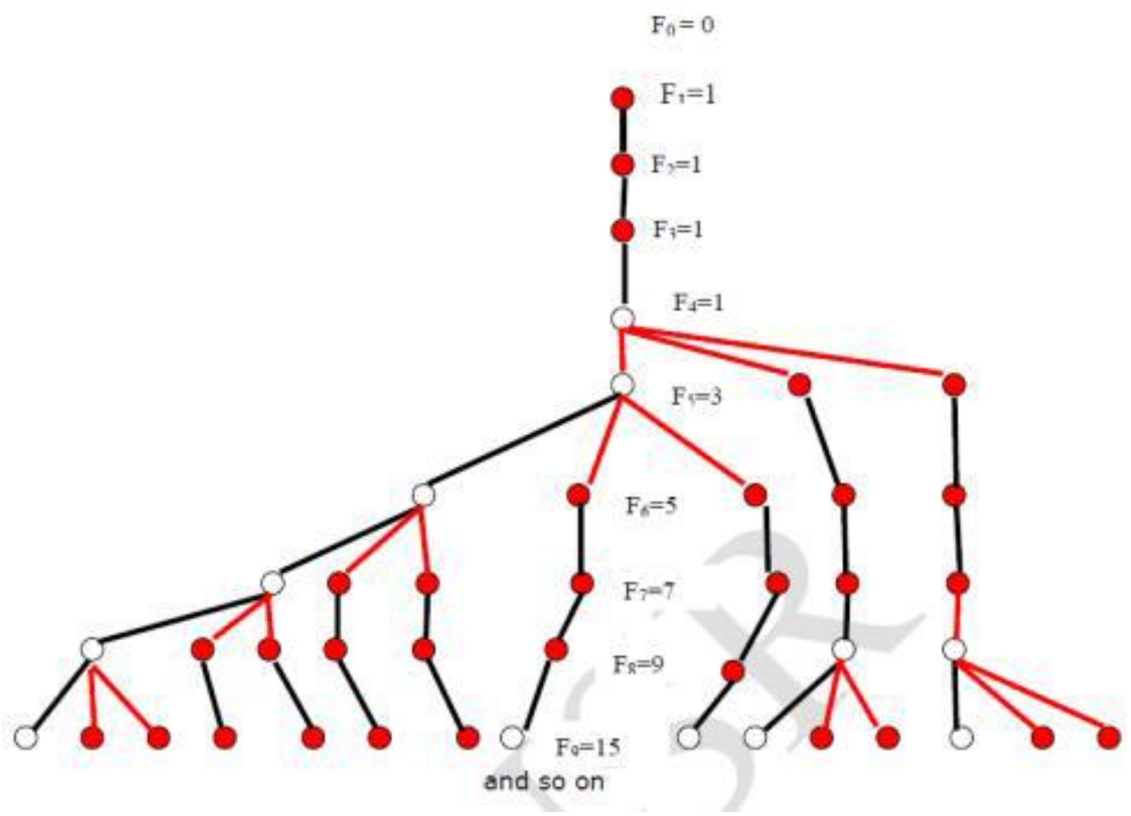
$$\begin{aligned} k_{(k+1)} + k_{(k+1)} + k_{(k+1)} &= (1\ (k-1))_{(k+1)} + k_{(k+1)} \\ &= (2\ (k-2))_{(k+1)} \\ &= (2\ ((k+1)-3))_{(k+1)} \end{aligned}$$

Hence proved.

Question:

Start with some special kind of pair of rabbits, one male and one female, born on January 1. Assume all months are of equal length and that rabbits begin to produce four months after their own birth. After reaching age of four months, each pair produces two mixed pairs, two males and two females, and then other two mixed pairs each month, and no rabbit dies.

1. Develop a model using tree diagram



2. Describe a recursive mathematical model

Total pairs at level k = Total pairs at level k-1 + Total pairs born at level k (1)

Since, Total pairs born at level k = 2×Total pairs at level k- 4 (2)

By (1) and (2)
Total pairs at level k = Total pairs at level k-1 + 2×Total pairs at level k-4

Now let us denote F_k = Total pairs at level k
Hence the recursive mathematical model would be

$$F_k = F_{k-1} + 2.F_{k-4}$$

3. Compute the number of pairs of rabbits after one year?

Computing number of pairs of rabbits after one year i.e. after 12 months

Since $F_k = F_{k-1} + 2F_{k-4}$ $F_0 = 0, F_1 = F_2 = F_3 = 1$

$F_4 = F_3 + 2F_0 = 1 + 2\times 0 = 1$
$F_5 = F_4 + 2F_1 = 1 + 2\times 1 = 3$

$F_6 = F_5 + 2F_2 = 3 + 2\times 1 = 5$
$F_7 = F_6 + 2F_3 = 5 + 2\times 1 = 7$
$F_8 = F_7 + 2F_4 = 7 + 2\times 1 = 9$
$F_9 = F_8 + 2F_5 = 9 + 2\times 3 = 15$
$F_{10} = F_9 + 2F_6 = 15 + 2\times 5 = 25$
$F_{11} = F_{10} + 2F_7 = 25 + 2\times 7 = 39$
$F_{12} = F_{11} + 2F_8 = 39 + 2\times 9 = 57$

Hence the number of pairs of rabbits after one year is = 57.

Question: Use Mathematical Induction to prove that sum of first m even positive integers is m(m+1)? OR Use Mathematical Induction to prove [2i = m(m+1)]

$$\sum_{i=1}^{i=m} 2i = m(m+1)$$

Instructor Solution

Solution:

Let P(n) denote the proposition

$$P(n) = \sum_{i=1}^{i=m} 2i = m(m+1)$$

Basic Step:
P(1) is true, since 2=2

Inductive step:
Let P(k) is true for positive integer k, i.e. 2+4+…..+(2k) = k(k+1)

Note that

$$\begin{aligned} 2+4+\ldots..+(2k)+(2k+2) &= k(k+1)+(2k+2) \\ &= k(k+1)+2(k+1) \\ &= (k+1)(k+2) \\ &= (k+1)((k+1)+1) \end{aligned}$$

So P(k+1) true, by induction, P(n) is true for all n belongs to positive integers.

My Solution from Assignment No: 1

$$\sum_{i=1}^{i=m} 2i = m(m+1)$$

Let compute m = 1 on both sides, we will get

$$\sum_{i=1}^{i=1} 2 * 1 = 1(1+1)$$

2 = 2 (True)

Therefore P (1) is true.

Now, suppose the statement is true for m = k

$$\sum_{i=1}^{k} 2i = k(k+1)$$

In order to prove it for m = k + 1, add 2(k+1) on both sides

$$\sum_{i=1}^{k} 2i + 2(k+1) = k(k+1) + 2(k+1)$$

By solving and taking k+1 as common, we will get

$$\sum_{i=1}^{k+1} 2i = (k+1)(k+2)$$

= (k+1) ((k+1) + 1)

= True

Hence, proved.

Question: Given two integers a and b, how to find their greatest common divisor (gcd(a, b))?

Proof:

Euclid's rule: If x and y are positive integers with $x \geq y$, then

$$\gcd(x, y) = \gcd(x \bmod y, y).$$

It is enough to show the slightly simpler rule

$\gcd(x, y) = \gcd(x - y, y)$.

Any integer that divides both x and y must also divide $x - y$, so

$\gcd(x, y) \leq \gcd(x - y, y)$.

Likewise, any integer that divides both $x - y$ and y must also divide both x and y, so $\gcd(x, y) \geq \gcd(x - y, y)$.

Question: Suppose sequence, $b_0, b_1, b_2, \ldots,$ satisfies Recurrence Relation

$$b_k = 6b_{k-1} - 9b_{k-2} \qquad \forall\ k \geq 2$$

With initial condition: $b_0 = 2$ and $b_1 = 6$

Then find explicit formula for $b_0, b_1, b_2, \ldots,$ using characteristic equation of the above recursion.

Solution:

$$b_k = 6b_{k-1} - 9b_{k-2}$$

or

$$b_k - 6b_{k-1} + 9b_{k-2} = 0$$

The characteristic equation for the above is given below

Let consider the followings

$b_k = x^2$

Therefore, $b_{k-1} = x$

Similarly, $b_{k-2} = x^0$

The characteristic equation will become as;

$x^2 - 6x + 9 = 0$

By simplifying

$(x)^2 – 2(x)(3) + (3)^2 = 0$

$(x – 3)^2 = 0$

$\Rightarrow t = 3$, is repeated root

Thus

3^n and $n.3^n$ are the sequences which satisfy the recurrence relation

Now, suppose general solution is:

$b_n = X.3^n + Z.n.3^n$

Which satisfies the original recurrence, here X and Z are constants.

By computing initial conditions: $b_0 = 2$ and $b_1 = 6$

For n = 0

$b_0 = X.3^0 + Z.0.3^0$

$2 = X + 0$ $[b_0 = 2]$

$\Rightarrow X = 2$ [i]

For n = 1

$b_1 = X.3^1 + Z.1.3^1$

$6 = 3X + 3Z$ $[b_1 = 6]$

$6 = 3(X + Z)$

$2 = X + Z$

$2 = 2 + Z$ [i]

$\Rightarrow Z = 0$

Thus,

$b_n = 2 . 3^n$ is the required explicit formula.

Question / Theorem: For any integer n ≥ 2, n is divisible by a prime.

Proof

By strong mathematical induction:

• Basis step:

The statement is true for n = 2. This is because 2 | 2 and 2 is a prime number.

• Inductive step:

Assume the statement is true for all i with $2 \leq i < k$ (inductive hypothesis) ;

To show that it is true for k

• We know that ∀ i Є Z, with $2 \leq i < k$, P(i), i.e. i is divisible by a prime number. (1)

• Now we show P(k), i.e., k is divisible by a prime.

Take two cases:

Case 1: k is prime.

Then k is divisible by itself. And nothing to prove

Case 2: k is composite.

Then $k = a \cdot b$, where $2 \leq a < k$ and $2 \leq b < k$

Based on (1), p|a for some prime p. (2)

Based on Case 2, a|k (3)

By transitivity, p | a and a | k ⇒ p|k

Thus, P(n) is true by strong induction.

Question: (Part-1) Use Mathematical Induction to prove that sum of the first m odd positive integers is m^2? OR Σ (2i - 1) = m^2 OR Σ (2i - 1) = n^2 (replace *m* with *n*)

Use Mathematical Induction to prove that sum of the first n odd positive integers is n^2?

Solution:

Let P(n) denote the proposition

$$P(n) = \sum_{i=1}^{i=m} (2i - 1) = m^2$$

Basic Step:
P(1) is true, since 2(1) = 1^2.

Inductive step:
Let P(k) is true for positive integer k, i.e. 1+3+5+…..+(2k-1) = k^2

Note that
1+3+5+…..+(2k-1) +(2k+1) = k^2+2k+1
= $(k+1)^2$

So P(k+1) true, by induction, P(n) is true for all n belongs to positive integers.

Question: (Part 2) Use Mathematical Induction to prove that sum of first m even positive integers is m(m+1)? **Σ 2i = m(m+1)**

[Note: Repeated Solution]

Solution:

Let P(n) denote the proposition

$$P(n) = \sum_{i=1}^{i=m} 2i = m(m+1)$$

Basic Step:
P(1) is true, since 2=2

Inductive step:
Let P(k) is true for positive integer k, i.e. 2+4+…..+(2k) = k(k+1)

Note that

$$\begin{aligned} 2+4+\ldots..+(2k) +(2k+2) &= k(k+1) + (2k+2) \\ &= k(k+1) + 2(k+1) \\ &= (k+1)\,(k+2) \\ &= (k+1)\,((k+1)+1) \end{aligned}$$

So P(k+1) true, by induction, P(n) is true for all n belongs to positive integers.

Question [Part 3]: Deduce the result of *Part 2* from *Part 1* and vice versa?

Solution:

Deduce the result of part 2 from part 1:

$$\sum_{i=1}^{i=m}(2i-1)=m^2$$

Adding 1 in each of the first m odd numbers gives us the first m even numbers, i.e.

$$\sum_{i=1}^{i=m}(2i-1+1)=m^2+m$$

$$\sum_{i=1}^{i=m}2i=m(m+1)$$

Similarly, vice versa

$$\sum_{i=1}^{i=m}2i=m(m+1)$$

Subtracting 1 from each of the first m even numbers gives us the first m odd numbers, i.e.

$$\sum_{i=1}^{i=m}(2i-1)=m(m+1)-m$$

$$\sum_{i=1}^{i=m}(2i-1)=m^2$$

Question: Use mathematical Induction to prove that the inequality

$n < 2^n$ for all n Є Z^+

<u>Proof:</u>

Let $P(n)$ be the proposition that $n < 2^n$

Basis step : $P(1)$ is true since $1 < 2^1$.

Inductive step :

Assume that $P(n)$ is true for a positive integer $n = k$, i.e., $k < 2^k$.

Now consider for $P(k+1)$:

Since, $k + 1 < 2^k + 1 \leq 2^k + 2^k = 2.2^k = 2^{k+1}$

$\therefore P(k+1)$ is true.

It proves that $P(n)$ is true for all $n \in Z^+$.

Question: Show by Mathematical Induction that any amount in cents ≥ 8 cents can be obtained using 3 cents and 5 cents coins only.

Proof

We have to prove that, amount = 3.m + 5.n, $m \geq 0, n \geq 0$

Basis Step

This time check for a five particular values:

$8 = 1.3 + 1.5$

$9 = 3.3$

$10 = 2.5$

$11 = 2.3 + 1.5$

$12 = 4.3$

Now we generalize it?

Let P(n) be the statement that:

"n cents can be obtained using 3 and 5 cents".

Inductive Hypothesis

We want to show that

$$P(k) \text{ is true} \Rightarrow P(k+1),\ \forall\, k \geq 8$$

There are two cases now

Case 1

P(k) is true and k cents contain at least one 5 coin.

Case 2

P(k) true, k cents do not contain any coin of 5 cent

Continue to next page….

Case 1

P(k) is true and k cents contain at least one 5 coin.

Since P(k) is true $k \geq 8$

Hence k can be expressed as

$k = 3.m + 5.n \quad m \geq 0 \text{ and } n \geq 1$

$k + 1= 3.m + 5.n + 1$

$k + 1= 3.m + 5.(n - 1) + 1 + 5$

$k + 1= 3.(m + 2) + 5.(n - 1), \quad m \geq 2 \text{ and } n \geq 0$

Hence the statement is true for $n = k + 1$

Case 2

- P(k) is true and k cents do not contain any coin of 5 cent. for $k \geq 8$

Hence k can be expressed as

$k = 3.m \quad m \geq 3$

$k + 1= 3.(m - 3) + 9 + 1$

$k + 1= 3.(m - 3) + 2.5$

$k + 1= 3.m' + 5.n \quad m' \geq 0 \text{ and } n = 2$

Hence the statement is true for $n = k + 1$

Hence P(k + 1) is true

Question: Prove that postage ticket of amount ≥ 12 cents can be formed using only 4 cent and 5 cent stamps.

Proof

Let $P(n) \equiv n$ cents can be formed using only 4 and 5 cent

$P(n) \equiv \mathrm{n} = 4\mathrm{s} + 5\mathrm{t} \qquad \mathrm{s} \geq 0$, and $\mathrm{t} \geq 0 \; \forall \; \mathrm{n} \geq 12$

Basis : $P(12)$ is true, since $12 = 4 \times 3$;

$P(13)$ is true, since $13 = 4 \times 2 + 5 \times 1$;

$P(14)$ is true, since $14 = 4 \times 1 + 5 \times 2$;

$P(15)$ is true, since $15 = 5 \times 3$;

Inductive : Assume $P(12), P(13), \ldots, P(k)$ are true.

Now prove for $P(k + 1)$

Suppose k-3 $= 4 \times s + 5 \times t$.

Then $k + 1 = 4 \times (s + 1) + 5 \times t$. true for $\mathrm{n} = \mathrm{k} + 1$.

By Strong Induction, $P(n)$ is true if $n \in \mathrm{Z}$ and $n \geq 12$.

Question

Show by mathematical induction that any amount in cents ≥ 18 cents can be obtained using 4 cents and 7 cents coins only.

Answer:

We have to prove that, amount = 4.m + n, $m \geq 0, n \geq 0$

Basis Step:

4*1 + 7*2 = 18

4*3 + 7*1 = 19

4*5 + 7*0 = 20

4*0 + 7*3 = 21

4*2 + 7*2 = 22

4*4 + 7*1 = 23

4*6 + 7*0 = 24

Let P(n) be the statement that: “n cents can be obtained using 4 and 7 cents”.

Inductive Hypothesis:

We want to show that

$P(k)$ is true $\Rightarrow P(k + 1), \forall k \geq 18$

Suppose $k - 3 = 4 * m + 7 * n$

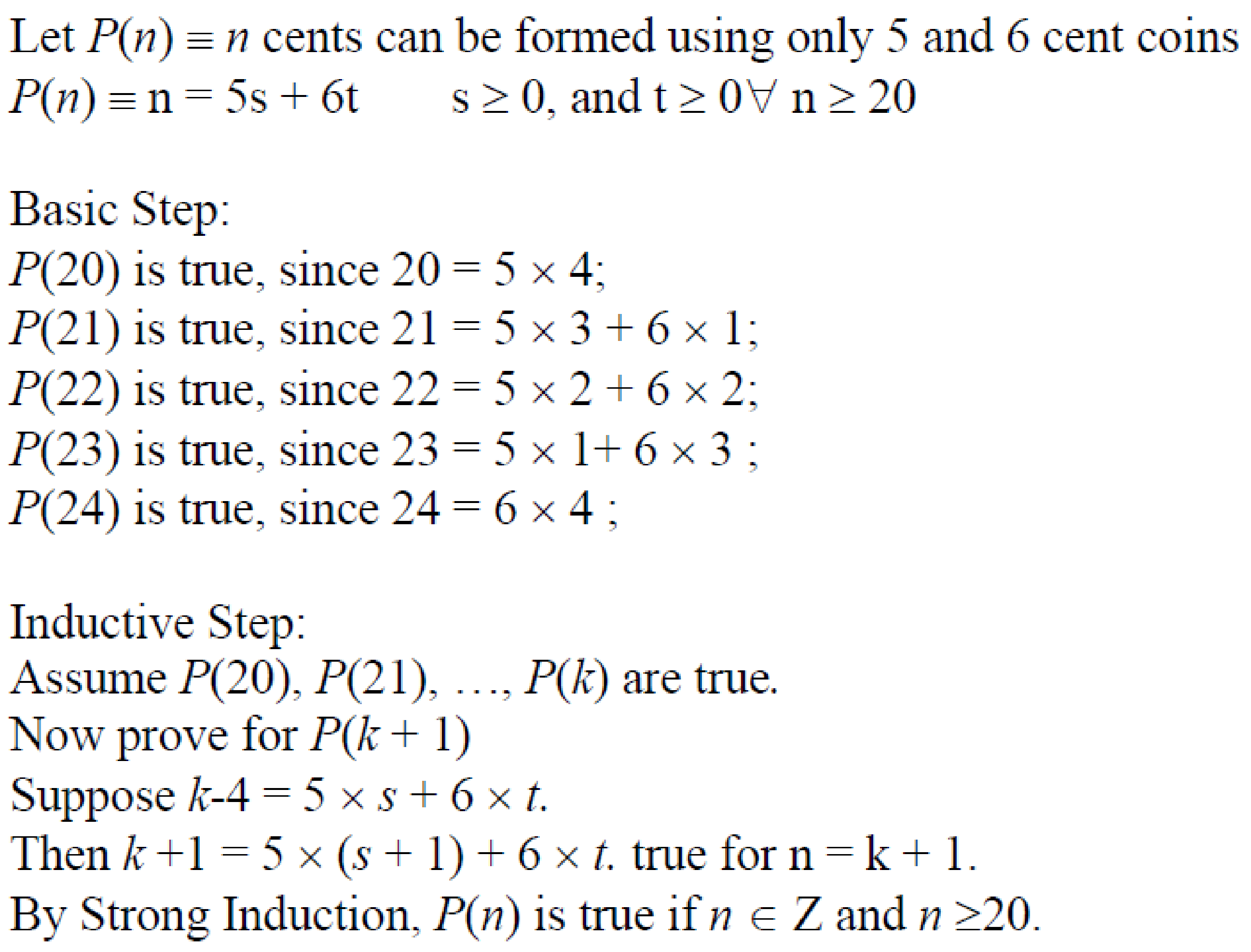
Then $k - 3 + 4 = 4 * m + 7 * n + 4$

$k + 1 = 4 * (m + 1) + 7 * n$

By Strong Induction, P(n) is true if n Є Z and n ≥18

Question: Show by mathematical induction that any amount in cents ≥ 20 cents can be obtained using 5 cents and 6 cents coins only?

Solution:



Let $P(n) \equiv n$ cents can be formed using only 5 and 6 cent coins
$P(n) \equiv n = 5s + 6t \qquad s \geq 0$, and $t \geq 0 \, \forall \, n \geq 20$

Basic Step:
$P(20)$ is true, since $20 = 5 \times 4$;
$P(21)$ is true, since $21 = 5 \times 3 + 6 \times 1$;
$P(22)$ is true, since $22 = 5 \times 2 + 6 \times 2$;
$P(23)$ is true, since $23 = 5 \times 1 + 6 \times 3$;
$P(24)$ is true, since $24 = 6 \times 4$;

Inductive Step:
Assume $P(20), P(21), \ldots, P(k)$ are true.
Now prove for $P(k + 1)$
Suppose k-4 $= 5 \times s + 6 \times t$.
Then $k + 1 = 5 \times (s + 1) + 6 \times t$. true for $n = k + 1$.
By Strong Induction, $P(n)$ is true if $n \in Z$ and $n \geq 20$.

Existence of Binary Integer Representation

Theorem:

Given any positive integer n, there exists a unique representation of n in the form:

$n = c_r . 2^r + c_{r-1} . 2^{r-1} + \dots + c_1 . 2^1 + c_0$

Where r is non-negative integer, $c_r = 1$, and $c_j = 0$ or 1, $\forall$ j = 0, 1, 2, . . . , r-1

Proof (*by strong induction*)

Let P(n) be the statement that n can be written in the form

$$n = c_r \cdot 2^r + c_{r-1} \cdot 2^{r-1} + \dots + c_1 \cdot 2^1 + c_0$$

Basis step:

If n = 1, then $n = c_r.2_r = c_0$, where r = 0, and $c_0 = 1$

Hence the statement is true for n = 1, i.e. P(1) is true

Inductive Hypothesis:

Let us suppose that statement is true for all i, $1 \leq i < k$,

$i = c_k.2^k + c_{k-1}.2^{k-1} + \dots + c_1.2^1 + c_0$
$c_r. = 1$, and $c_i = 0$ or 1, $\forall$ j = 0, 1, 2, . . . , r-1

Show that

Continue Next Page…

Now we prove that statement is true for k

<u>Case 1</u>

Suppose k is even, k/2 is an integer and k/2 < k, hence

$k/2 = c_r.2^r + c_{r-1}.2^{r-1} + \ . . . + c_1.2^1 + c_0$

where r is non-negative integer and

$c_r = 1$, and $c_i = 0$ or 1, $\forall\ j = 0, 1, 2, . . . , r-1$

$k = 2.c_r.2^r + 2.c_{r-1}.2^{r-1} + \ . . . + 2.c_1.2^1 + 2.c_0$

$k = cr.2^{r+1} + c_{r-1}.2_r + \ . . . + c_1.2_2 + c_0.2^1$, true

which is the required form

<u>Case 2</u>

Let $k \geq 3$, is odd, $(k-1)/2$ is an integer and $1 \leq (k-1)/2 < k$,

$(k-1)/2 = c_r.2_r + c_{r-1}.2_{r-1} + \ . . . + c_1.2_1 + c_0$

where r is non-negative integer and

$c_r = 1$, and $c_j = 0$ or 1, $\forall\ j = 0, 1, 2, . . . , r-1$

Now, $k - 1 = c_r.2^{r+1} + c_{r-1}.2^r + \ . . . + c_1.2^2 + c_0.2^1$

And, $k = c_r.2_{r+1} + c_{r-1}.2^r + \ . . . + c_1.2^2 + c_0.2^1 + 1$, true

Hence by strong mathematical induction, P(n) is true

Fibonacci Sequences

Computing Values using Mathematical Model

Since $F_k = F_{k-1} + F_{k-2}$ $F_0 = 0, F_1 = 1$

F2 = F1 + F0= 1 + 0 = 1

F3 = F2 + F1= 1 + 1 = 2

F4 = F3 + F2= 2 + 1 = 3

F5 = F4 + F3= 3 + 2 = 5

F6 = F5 + F4= 5 + 3 = 8

F7 = F6 + F5= 8 + 5 = 13

F8 = F7 + F6= 13 + 8 = 21

F9 = F8 + F7= 21 + 13 = 34

F10 = F9 + F8= 34 + 21 = 55

F11 = F10 + F9= 55 + 34 = 89

F12 = F11 + F10= 89 + 55 = 144 . . .

Rabbits Problem

Refer to Handouts: CS702 Handouts_MidTerm_Genrica

Statement	Page No
After reaching age of **two months** , each pair produces two other mixed pairs, two male and two female	35 to 37
After reaching age of **three months**, each pair produces another mixed pairs, one male and other female, and then another mixed pair each month	37 to 38

Within This File: Scroll to Page No 71

Statement	Page No
Begin to produce **four months** after their own birth. After reaching age of four months, each pair produces two mixed pairs.	71 to 72

Applications of Fibonacci Sequences

Fibonacci sequences are used

- In trend analysis
- By some pseudorandom number generators
- The number of petals is a Fibonacci number.
- Many plants show the Fibonacci numbers in the arrangements of the leaves around the stems.
- Seen in arrangement of seeds on flower heads
- Consecutive Fibonacci numbers give worst case behavior when used as inputs in Euclid's algorithm.
- As n approaches infinity, the ratio F(n+1)/F(n) approaches the golden ratio: Φ =1.6180339887498948482...
- The Greeks felt that rectangles whose sides are in the golden ratio are most pleasing
- The Fibonacci number F(n+1) gives the number of ways for 2 x 1 dominoes to cover a 2 x n checkerboard.
- Sum of the first n Fibonacci numbers is F(n+2)-1.
- The shallow diagonals of Pascal's triangle sum to Fibonacci numbers.
- Except n = 4, if F(n) is prime, then n is prime.
- Equivalently, if n not prime, then F(n) is not prime.
- gcd(F(n), F(m)) = F(gcd(n, m))

What is Recursion?

- Sometimes problems are too difficult or too complex to solve because these are too big.
- A problem can be broken down into sub-problems and find a way to solve these sub-problems
- Then build up to a solution to the entire problem.
- This is the idea behind recursion
- Recursive algorithms break down problem in pieces which you either already know the answer, or can be solved applying same algorithm to each piece
- And finally combine the results of these sub-problems to find the final solution
- More concisely, a recursive function or definition is defined in terms of itself.
- Recursion is a computer algorithm that calls itself in a steps having a termination condition.
- The successive repetitions are processed up to the critical step where the condition is met.
- In recursive algorithms, each repetition is processed from the last one called to the first.
- Recursion is a wonderful technique for dealing with many problems where problems and sub-problems have same mechanism to solve it.
- Merits and Demerits of Recursion
- Recursive solutions are much easier to conceive of and code than their iterative counterparts.
- Every problem is not solvable using this approach
- What kinds of problems are solved with recursion?
- Generally, problems which are defined in terms of themselves are usually good candidates for recursive techniques.

Question: Algorithm: Maxima Finding Problem

Solution:

Input: A set S of 2-dimensional points.

Output: The maximal set of S.

```
Maxima(P[1..n])
  1. Sort the points in ascending order w. r .t. X axis
  2. If |S| = 1, then return it, else
       find a line perpendicular to X-axis which separates S into $S_L$ and $S_R$, each of which
       consisting of n/2 points.
  3. Recursively find the maxima's $S_L$ and $S_R$
  4. Project the maxima's of $S_L$ and $S_R$ onto L and sort these points according to their y-
     values.
  5. Conduct a linear scan on the projections and discard each of maxima of $S_L$ if its y-value
     is less than the y-value of some maxima's of $S_R$.
```

Time Complexity

$$T(n) = \begin{cases} 2T(n/2) + O(n) + O(n) & , \quad n \geq 2 \\ 1 & , \quad n < 2 \end{cases}$$

Assume n = 2k, then

$$\begin{aligned} T(n) &= 2T(n/2) + n + n \\ &= 2(2T(n/4) + n/2 + n/2) + n + n \\ &= 2^2T(n/2^2) + n + n + n + n \\ &= 2^2T(n/2^2) + 4n \\ &= 2^2(2T(n/2^3) + n/4 + n/4) + 4n \\ &= 2^3T(n/2^3) + n + n + 6n \end{aligned}$$

$$T(n) = 2^3T(n/2^3) + n + n + 6n$$

.

.

$$T(n) = 2^kT(n/2^k) + 2kn$$

$$= 2^kT(2^k/2^k) + 2kn \quad \text{Since } n = 2^k$$

Hence

$$T(n) = 2k + 2kn$$

$$T(n) = 2k + 2kn \qquad n = 2^k \Rightarrow k = \log(n)$$

$$T(n) = n + 2n.\log n = \Theta(n.\log n)$$

Merge Sort

Merge-sort is based on divide-and-conquer approach and can be described by the following three steps:

Divide Step:

- If given array A has zero or one element, return S.
- Otherwise, divide A into two arrays, A1 and A2,
- Each containing about half of the elements of A.

Recursion Step:

- Recursively sort array A1, A2

Conquer Step:

- Combine the elements back in A by merging the sorted arrays A1 and A2 into a sorted sequence.

Visualization of Merge-sort as Binary Tree

- We can visualize Merge-sort by means of binary tree where each node of the tree represents a recursive call
- Each external node represents individual elements of given array A.
- Such a tree is called Merge-sort tree.
- The heart of the Merge-sort algorithm is conquer step, which merge two sorted sequences into a single sorted sequence

Merge Sort Algorithm

```
Merge-sort(A, f, l)
  1.        if f < l
  2.            then m = (f + l)/2
  3.            Merge-sort(A, f, m)
  4.            Merge-sort(A, m + 1, l)
  5.            Merge(A, f, m, l)
```

Merge-sort Algorithm

```
Merge(A, f, m, l)
  1.  T[f..l]                     \\declare temporary array of same size
  2.  i ← f; k ← f; j ← m + 1      \\initialize integers i, j, and k
  3.  while (i ≤ m) and (j ≤ l)
  4.  do if (A[i] ≤ A[j])          \\comparison of elements
  5.       then T[k++] ← A[i++]
  6.       else  T[k++] ← A[j++]
  7.  while (i ≤ m)
  8.  do  T[k++] ← A[i++]          \\copy from A to T
  9.  while (j ≤ l)
  10. do  T[k++] ← A[j++]          \\copy from A to T
  11. for i ← p to r
  12. do A[i] ← T[i]               \\copy from T to A
```

Analysis of Merge-sort Algorithm

- Let $T(n)$ be the time taken by this algorithm to sort an array of n elements dividing A into sub-arrays A_1 and A_2.
- It is easy to see that the Merge (A_1, A_2, A) takes the linear time. Consequently,

$$T(n) = T(n/2) + T(n/2) + \theta(n)$$

$$T(n) = 2T\ (n/2) + \theta(n)$$

- The above recurrence relation is non-homogenous and can be solved by any of the methods
  - Defining characteristics polynomial
  - Substitution
  - recursion tree or
  - master method

Algorithm: Maxima Finding Problem

Input: A set S of 2-dimensional points.
Output: The maximal set of S.

Maxima(P[1..n])
1. Sort the points in ascending order w. r .t. X axis
2. If |S| = 1, then return it, else
    Find a line perpendicular to X-axis which separates S into S_L and S_R, each of which consisting of n/2 points.
3. Recursively fined the maxima's S_L and S_R
4. Project the maxima's of S_L and S_R onto L and sort these points according to their y-values.
5. Conduct a linear scan on the projections and discard each of maxima of S_L if its y-value is less than the y-value of some maxima's of S_R.

Time Complexity

$$T(n) = \begin{cases} 2T(n/2) + O(n) + O(n) & , \quad n \geq 2 \\ 1 & , \quad n < 2 \end{cases}$$

Assume n = 2k, then

$$\begin{aligned} T(n) &= 2T(n/2) + n + n \\ &= 2(2T(n/4) + n/2 + n/2) + n + n \\ &= 2^2T(n/2^2) + n + n + n + n \\ &= 2^2T(n/2^2) + 4n \\ &= 2^2(2T(n/2^3) + n/4 + n/4) + 4n \\ &= 2^3T(n/2^3) + n + n + 6n \end{aligned}$$

$$T(n) = 2^3T(n/2^3) + n + n + 6n$$

.

.

$$\begin{aligned} T(n) &= 2^kT(n/2^k) + 2kn \\ &= 2^kT(2^k/2^k) + 2kn \quad \text{Since } n = 2^k \end{aligned}$$

Hence

$$T(n) = 2k + 2kn$$

$$T(n) = 2k + 2kn \qquad n = 2^k \Rightarrow k = \log(n)$$

$$T(n) = n + 2n.\log n = \Theta(n.\log n)$$

Question: Why the Optimal Weight Convex Polygon Problem cannot be solved using Divide and Conquer Approach?

Solution:

Optimal Weight Convex polygon problem cannot be solved using divide and conquer approach because:

- Its sub problems are not independent from each other.
- We need an optimal solution of the problem, which may not be found using divide and conquer approach.

Question: Consider the problem of a chain of A$_1$, A$_2$ and A$_3$ of three matrices, suppose the dimensions of the matrices are 10x100, 100x5, and 5x50. For the sequence of matrices given above compute the order of product A$_1$, A$_2$ and A$_3$ is such a way that minimum total number of scalar multiplications.

Solution:

Order of A$_1$= 10 x 100
Order of A$_2$= 100 x 5
Order of A$_3$= 5 x 50

Given sequence of matrices

$$\frac{A_1}{10\times100}\cdot\frac{A_2}{100\times5}\cdot\frac{A_3}{5\times50}$$

$P_0 = 10$, $P_1 = 100$, $P_2 = 5$, $P_3 = 50$

m[1,1]	m[1,2]	m[1,3]
	m[2,2]	m[2,3]
		m[3,3]

<u>Main Diagonal</u>

$$m[i,i] = 0,\ \forall i = 1,2,3$$

$$m[i,j] = \min_{i\le k<j}\left(m[i,k] + m[k+1,j] + p_{i-1}.p_k.p_j\right)$$

$m[1,1] = 0$, $m[2,2] = 0$, $m[3,3] = 0$

Computing m[1,2] and m[2,3]

Computing m[1,2]

$$m[i,j] = \min_{i \le k < j}\left(m[i,k] + m[k+1,j] + p_{i-1}.p_k.p_j\right)$$

$$m[1,2] = \min_{1 \le k < 2}\left(m[1,k] + m[k+1,2] + p_{1-1}.p_k.p_2\right)$$

Putting k = 1

$$m[1,2] = \min_{1 \le k < 2}\left(m[1,1] + m[2,2] + p_0.p_1.p_2\right)$$

$$m[1,2] = \min_{1 \le k < 2}\left(0 + 0 + 10 \cdot 100 \cdot 5\right)$$

$$m[1,2] = 5,000$$

$$s[1,2] = k = 1$$

Computing m[2,3]

$$m[i,j] = \min_{i \le k < j}\left(m[i,k] + m[k+1,j] + p_{i-1}.p_k.p_j\right)$$

$$m[2,3] = \min_{2 \le k < 3}\left(m[2,k] + m[k+1,3] + p_{2-1}.p_k.p_3\right)$$

Putting k = 2

$$m[2,3] = \min_{2 \le k < 3}\left(m[2,2] + m[3,3] + p_1.p_2.p_3\right)$$

$$m[2,3] = \min_{2 \le k < 3}\left(0 + 0 + 100 \cdot 5 \cdot 50\right)$$

$$m[2,3] = 25,000$$

$$s[2,3] = k = 2$$

Computing m[1,3]

$$m[i,j] = \min_{i \le k < j}\left(m[i,k] + m[k+1,j] + p_{i-1}.p_k.p_j\right)$$

$$m[1,3] = \min_{1 \le k < 3}\left(m[1,k] + m[k+1,3] + p_{1-1}.p_k.p_3\right)$$

Putting k = 1 and k = 2

$$m[1,3] = \min_{1 \le k < 3}\left(\left(m[1,1] + m[2,3] + p_0.p_1.p_3\right), \left(m[1,2] + m[3,3] + p_0.p_2.p_3\right)\right)$$

$$m[1,3] = \min_{1 \le k < 3}\left(\left(0 + 25,000 + 10 \cdot 100.50\right), \left(5,000 + 0 + 10 \cdot 5 \cdot 50\right)\right)$$

$$m[1,3] = \min_{1 \le k < 3}\left(75000, 7500\right)$$

$$m[1,3] = 7500$$

$$s[1,3] = k = 2$$

Final Cost Matrix and its Order of Computation

Final Cost Matrix

0	5000	7500
	0	25,000
		0

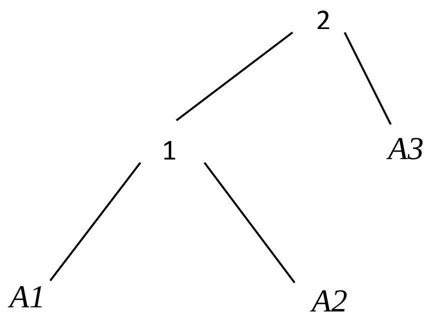
Order of Computation

1	4	6
	2	5
		3

k’s Values Leading to - Minimum m[i,j]

0	1	2
	0	2
		0

Order of Computation will be $(A_1.A_2).A_3$



Question: For the sequence of matrices, given below, compute the order of the product, $A_1.A_2.A_3.A_4$, in such a way that minimizes the total number of scalar multiplications, using Dynamic Programming.

Order of A_1 = 20 x 10

Order of A_2 = 10 x 30

Order of A_3 = 30 x 15

Order of A_4 = 15 x 25

Solution:

Computing the optimal multiplication order for a series of matrices

$$\frac{A1}{20 \times 10} \cdot \frac{A2}{10 \times 30} \cdot \frac{A3}{30 \times 15} \cdot \frac{A4}{15 \times 25}$$

From above we have
$P_0 = 20$
$P_1 = 10$
$P_2 = 30$
$P_3 = 15$
$P_4 = 25$

m[1,1]	m[1,2]	m[1,3]	m[1,4]
x	m[2,2]	m[2,3]	m[2,4]
x	x	m[3,3]	m[3,4]
x	x	x	m[4,4]

Mathematical Model of dynamic programming

$$m[i,i] = 0 \qquad \forall\ i = 1,2,3,4$$

$$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$$

Main Diagonals *m[i,i]*:

m[1, 1] = 0 m[2, 2] = 0 m[3, 3] = 0 m[4, 4] = 0

First Diagonals: m[1, 2], m[2, 3], m[3, 4]

Computing m[1, 2] i.e. i = 1 and j = 2:

$$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$$

$$m[1, 2] = \min_{1 \le k < 2} (m[1,k] + m[k+1, 2] + P_0 . P_k . P_2)$$

For k = 1, we will get

$$m[1, 2] = \min_{1 \le k < 2} (m[1,1] + m[2, 2] + P_0 . P_1 . P_2)$$

Substitute the known values

$m[1, 2] = \min_{1 \le k < 2} (0 + 0 + 20 . 10 . 30)$

$m[1, 2] = \min_{1 \le k < 2} (6000)$

$m[1, 2] = 6000$ **$s[1, 2] = k = 1$**

Computing m[2, 3] i.e. i = 2 and j = 3:

$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$

$m[2, 3] = \min_{2 \le k < 3} (m[2,k] + m[k+1, 3] + P_1 . P_k . P_3)$

For k = 2, we will get

$m[2, 3] = \min_{2 \le k < 3} (m[2,2] + m[3, 3] + P_1 . P_2 . P_3)$

Substitute the known values

$m[2, 3] = \min_{2 \le k < 3} (0 + 0 + 10 . 30 . 15)$

$m[2, 3] = 4500$ **$s[2, 3] = k = 2$**

Computing m[3, 4] i.e. i = 3 and j = 4:

$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$

$m[3, 4] = \min_{3 \le k < 4} (m[3,k] + m[k+1, 4] + P_2 . P_k . P_4)$

For k = 3, we will get

$m[3, 4] = \min_{3 \le k < 4} (m[3,3] + m[4, 4] + P_2 . P_3 . P_4)$

Substitute the known values

$m[3, 4] = \min_{k = 3} (0 + 0 + 30 . 15 . 25)$

$m[3, 4] = 11250$ **$s[3, 4] = k = 3$**

Second Diagonals: m[1, 3], m[2, 4]

Computing m[1, 3] i.e. i = 1 and j = 3:

$m[i,j] = min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$

$m[1, 3] = min_{1 \le k < 3} (m[1,k] + m[k+1, 3] + P_0 . P_k . P_3)$

For k = 1 and k = 2 we will get

$m[1, 3] = min_{1 \le k < 3} (\quad m[1,1] + m[2, 3] + P_0 . P_1 . P_3 ,$

$m[1,2] + m[3, 3] + P_0 . P_2 . P_3)$

Substitute the known values

$m[1, 3] = min_{1 \le k < 3} (\quad 0 + 4500 + 20 . 10 . 15 ,$

$6000 + 0 + 20 . 30 . 15)$

$m[1, 3] = min_{1 \le k < 3} (7500 , 15000)$

m[1, 3] = 7500 **s[1, 3] = k = 1**

Computing m[2, 4] i.e. i = 2 and j = 4:

$m[i,j] = min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$

$m[2, 4] = min_{2 \le k < 4} (m[2,k] + m[k+1, 4] + P_1 . P_k . P_4)$

For k = 2 and k = 3 we will get

$m[2, 4] = min_{2 \le k < 4} (\quad m[2,2] + m[3, 4] + P_1 . P_2 . P_4,$

$m[2,3] + m[4, 4] + P_1 . P_3 . P_4)$

Substitute the known values

$m[2, 4] = min_{2 \le k < 4} (\quad 0 + 11250 + 10 . 30 . 25,$

$4500 + 0 + 10 . 15 . 25)$

$m[2, 4] = min_{2 \le k < 4} (18750 , 8250)$

m[2, 4] = 8250 **s[2, 4] = k = 3**

Third Diagonal: m[1, 4]

Computing m[1, 4] i.e. i = 1 and j = 4:

$$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$$

$$m[1, 4] = \min_{1 \le k < 4} (m[1,k] + m[k+1, 4] + P_0 . P_k . P_4)$$

For k = 1, k = 2 and k = 3 we will get

$$m[1, 4] = \min_{1 \le k < 4} (m[1,1] + m[2, 4] + P_0 . P_1 . P_4 ,\quad m[1,2] + m[3, 4] + P_0 . P_2 . P_4 ,\quad m[1,3] + m[4, 4] + P_0 . P_3 . P_4)$$

Substitute the known values

$$m[1, 4] = \min_{1 \le k < 4} (0 + 8250 + 20 . 10 . 25 ,\quad 6000 + 11250 + 20 . 30 . 25 ,\quad 7500 + 0 + 20 . 15 . 25)$$

$$m[1, 4] = \min_{1 \le k < 4} (13250 , 32250 , 15000)$$

m[1, 4] = 13250 **s[1, 4] = k = 1**

Final Cost Matrix and Order of Computation

The procedure is as follows

- The table *m[1. . n, 1. . n]* for storing the *m[i, j]* costs
- The table s[1. . n - 1, 2. . n] records for which index of *k* achieved the optimal cost in computing *m[i, j]*
- The table *s* is used to construct an optimal solution

Final Cost Matrix

0	6000	7500	13250
	0	4500	8250
		0	11250
			0

Order of Computation

1	5	8	10
	2	6	9
		3	7
			4

Values of *k's* Leading to Minimum *m[i,j]*

0	1	1	1
	0	2	3
		0	3
			0

From above final cost matrix, the computation shows that the minimum cost for multiplying $A_1.A_2.A_3.A_4$ matrices, using Dynamic Programming is **13250** i.e. m[1,4].

Order of Computation to minimize the total number of scalar multiplications:

The optimal order for multiplication is $\mathbf{A_1 . ((A_2 . A_3) . A_4)}$

Question: Consider the chain matrix multiplication for 4 matrices:

A1	A2	A3	A4
(7×5)	(5×4)	(4×6)	(6×8)

Compute the cost table m in the dynamic programming algorithm for the Chain Matrix Multiplication.

Solution:

$$\frac{A1}{7\ x\ 5} \cdot \frac{A2}{5\ x\ 4} \cdot \frac{A3}{4\ x\ 6} \cdot \frac{A4}{6\ x\ 8}$$

From above we have
$P_0 = 7$
$P_1 = 5$
$P_2 = 4$
$P_3 = 6$
$P_4 = 8$

m[1,1]	m[1,2]	m[1,3]	m[1,4]
x	m[2,2]	m[2,3]	m[2,4]
x	x	m[3,3]	m[3,4]
x	x	x	m[4,4]

Mathematical Model of Dynamic Programming

$m[i,i] = 0 \qquad \forall\ i = 1,2,3,4$

$m[i,j] = \min_{i \le k < j} (m[i,k] + m[k+1, j] + P_{i-1} . P_k . P_j)$

First super diagonal

m[1,2]=m[1,1] + m[2,2] + p0.p1.p2 = 0+0+7.5.4 = 140

m[2,3]=m[2,2] + m[3,3] + p1.p2.p3 = 0+0+5.4.6= 120

m[3,4]=m[3,3] + m[4,4] + p2.p3.p4 = 0+0+4.6.8 = 192

Second super diagonal

m[1,3]=m[1,1] + m[2,3] + p0.p1.p3 = 0+120+7*5*6 = 330

m[1,3]=m[1,2] + m[3,3] + p0.p2.p3 = 140+0+7*4*6 = 308

Minimum [1,3] = 308

for m[2,4]

m[2,4]=m[2,2] + m[3,4] + p1.p2.p4 = 0+192+5*4*8 =352

m[2,4]=m[2,3] + m[4,4] + p1.p3.p4 = 120+0+5*6*8 = 360

Minimum for m[2,4] = 352

Third super diagonal

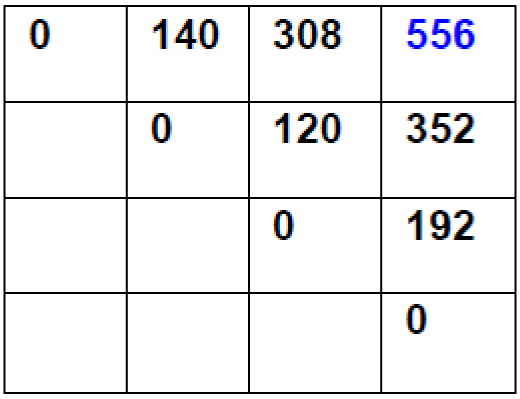
m[1,4]=m[1,1] + m[2,4] + p0.p1.p4 = 0+352+7*5*8 = 632

m[1,4]=m[1,2] + m[3,4] + p0.p2.p4 = 140+192+7*4*8 = 556

m[1,4]=m[1,3] + m[4,4] + p0.p3.p4 = 308+0+7*6*8 =644

Minimum for m[1,4] = 556

Resultant is,



0	140	308	556
	0	120	352
		0	192
			0

Question: Complexity function and j > k where j , k are numbers greater than 2. Every function is separated by “comma” and note that there are 20 functions to arrange.

$n\log n/10000$, $1000n^{6k/2}$, $n^{24j/4}$,
$\sqrt{10000}\lg n$, n^{n-1}, 52000000000, 2^n ,
$\sqrt[8]{n}\ \lg n$, $n!/100$, $(2^n n\sqrt{n})/\sqrt{n^2}$, $n!/\sqrt{n}$,
$2^n n\log n$, $n!/\log n$, 5000, $n^4/\sqrt[4]{n}$, $n(\log n)\sqrt[5]{n}$,
$n^7/\sqrt[11]{n}$, $n(\log n)\sqrt[3]{n}$, k^{n-2}, j^{n-4}

Solution:

3. $n!/\log n$
4. $n!/\sqrt{n}$
5. j^{n-4}
6. k^{n-2}
7. $2^n n\log n$
8. $(2^n n\sqrt{n})/\sqrt{n^2}$
9. 2^n
10. $n^{24j/4}$
11. $1000n^{6k/2}$
12. $n^7/\sqrt[11]{n}$
13. $n^4/\sqrt[4]{n}$
14. $n(\log n)\sqrt[3]{n}$
15. $n(\log n)\sqrt[5]{n}$
16. $n\log n/10000$
17. $\sqrt[8]{n}\ \lg n$
18. $\sqrt{10000}\lg n$
19. 52000000000
20. 5000

General Knapsack Problem

- Given a set of items, each with a cost and a value, then determine the items to include in a collection so that the total cost is less than some given cost and the total value is as large as possible.
- Knapsack problem is of combinatorial optimization
- It derives its name from the maximization problem of choosing possible essentials that can fit into one bag, of maximum weight, to be carried on a trip.
- A similar problem very often appears in business, complexity theory, cryptography and applied mathematics.

0-1 Knapsack Problem Statement

The knapsack problem arises whenever there is resource allocation with no financial constraints

Problem Statement

A thief robbing a store and can carry a maximal weight of W into his knapsack. There are n items and ith item weight is w_i and worth is v_i dollars. What items should thief take, not exceeding the bag capacity, to maximize value?

Assumption:

The items may not be broken into smaller pieces, so thief may decide either to take an item or to leave it, but may not take a fraction of an item.

Notations: 0-1 Knapsack Problem Construction

- You have prepared a list of n objects for which you are interested to buy, The items are numbered as $i_1, i_2, . . ., i_n$
- Capacity of bag is W
- Each item i has value vi, and weigh wi
- We want to select a set of items among $i_1, i_2, . . ., i_n$ which do not exceed (in total weight) capacity W of the bag
- Total value of selected items must be maximum
- How should we select the items?

0-1 Knapsack Algorithm

Question: Brute Force Algorithm for 0-1 Knapsack problem?

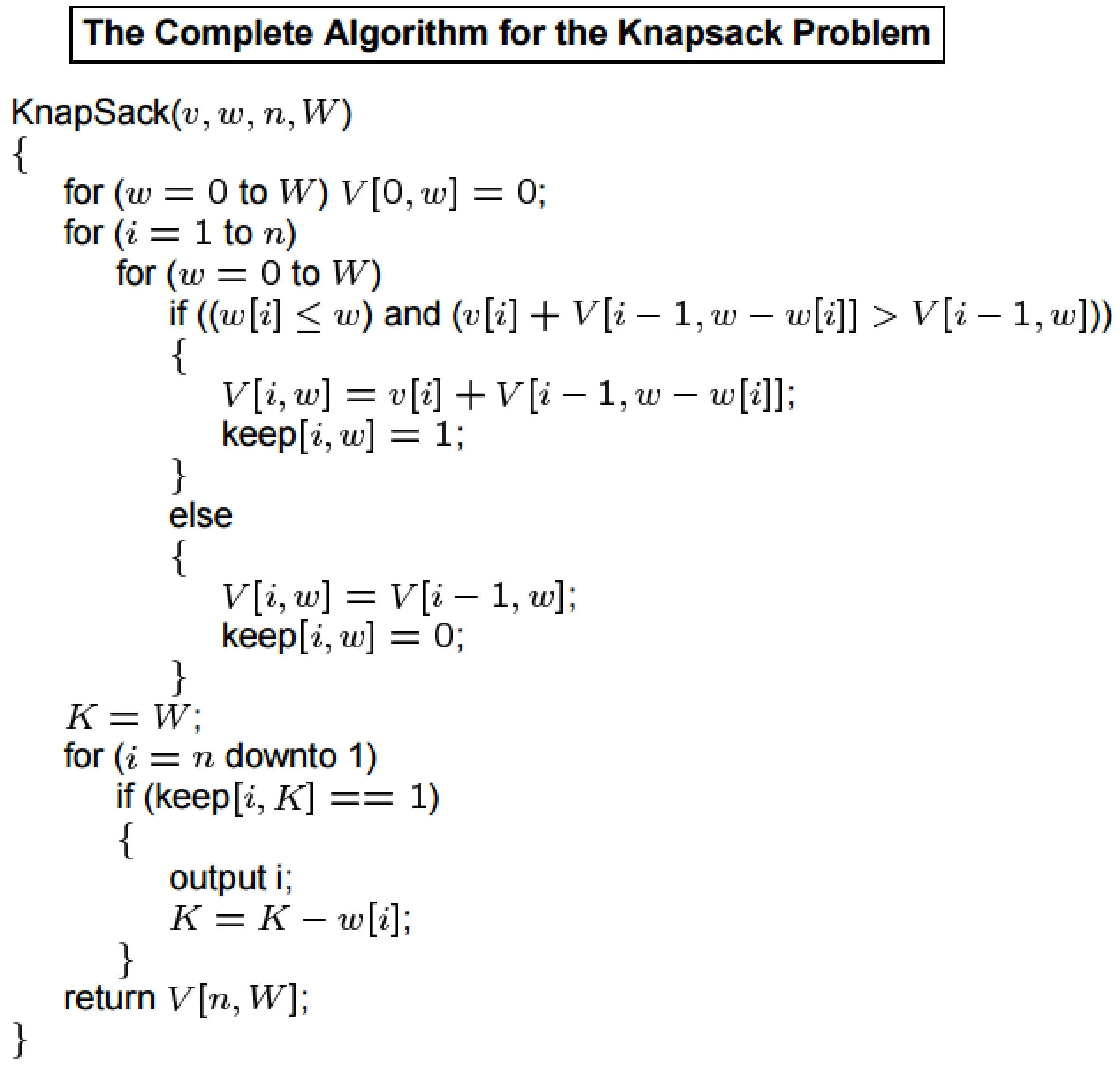
```
Knapsack-BF (n, V, W, C)
    Compute all subsets, s, of S = {1, 2, 3, 4}
    For all s ∈ S
      weight = Compute sum of weights of these items
          if weight > C, not feasible
      new solution = Compute sum of values of these items
      solution = solution ∪ {new solution}
    Return maximum of solution
```

0-1 Knapsack Algorithm Analysis

Approach
- In brute force algorithm, we go through all combinations and find the one with maximum value and with total weight less or equal to $W = 16$

Complexity
- Cost of computing subsets $O(2^n)$ for n elements
- Cost of computing weight = $O(2^n)$
- Cost of computing values = $O(2^n)$
- Total cost in worst case: $O(2^n)$

Question: Complete Dynamic Programming / Algorithm for 0-1 Knapsack problem



The Complete Algorithm for the Knapsack Problem

```
KnapSack($v, w, n, W$)
{
    for ($w = 0$ to $W$) $V[0, w] = 0$;
    for ($i = 1$ to $n$)
        for ($w = 0$ to $W$)
            if (($w[i] \leq w$) and ($v[i] + V[i-1, w-w[i]] > V[i-1, w]$))
            {
                $V[i, w] = v[i] + V[i-1, w-w[i]]$;
                keep$[i, w] = 1$;
            }
            else
            {
                $V[i, w] = V[i-1, w]$;
                keep$[i, w] = 0$;
            }
    $K = W$;
    for ($i = n$ downto 1)
        if (keep$[i, K]$ == 1)
        {
            output i;
            $K = K - w[i]$;
        }
    return $V[n, W]$;
}
```

Time complexity: Clearly, O(nW).

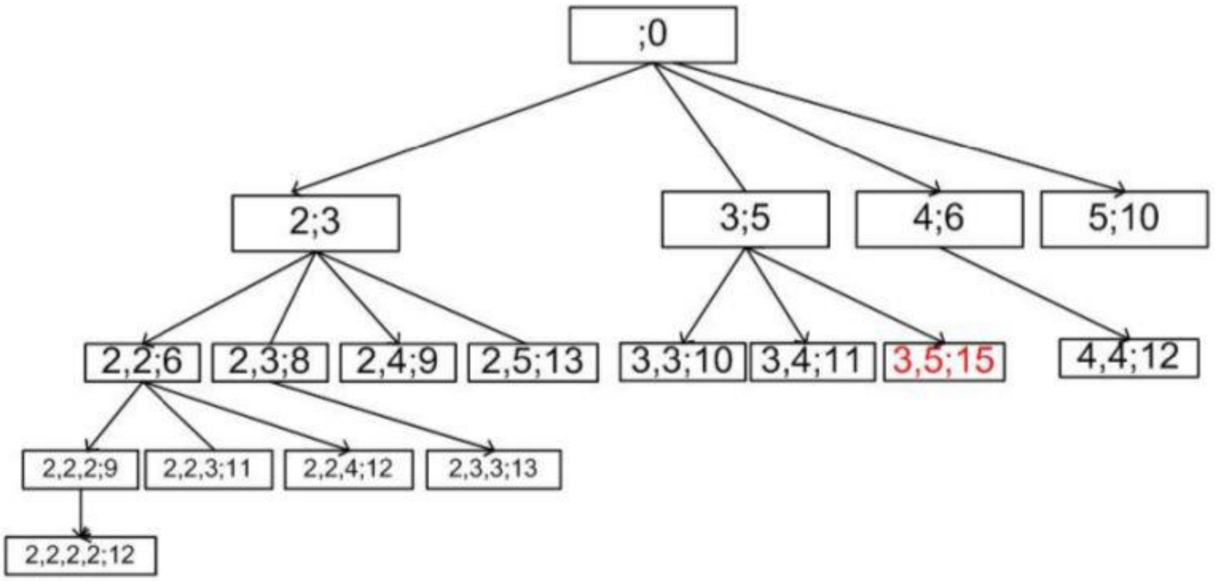
Question: Use the Backtracking algorithm for the 0-1 Knapsack problem to maximize the profit for the following problem instance. Show actions step by steps. Maximum Capacity = 8

Knapsack Example: Backtracking

Maximum Capacity = 8

i	1	2	3	4
v_i	3	5	6	10
w_i	2	3	4	5

(2,2,3;11) means that two elements of each weight 2 and one element of weight 3 is with total value 11



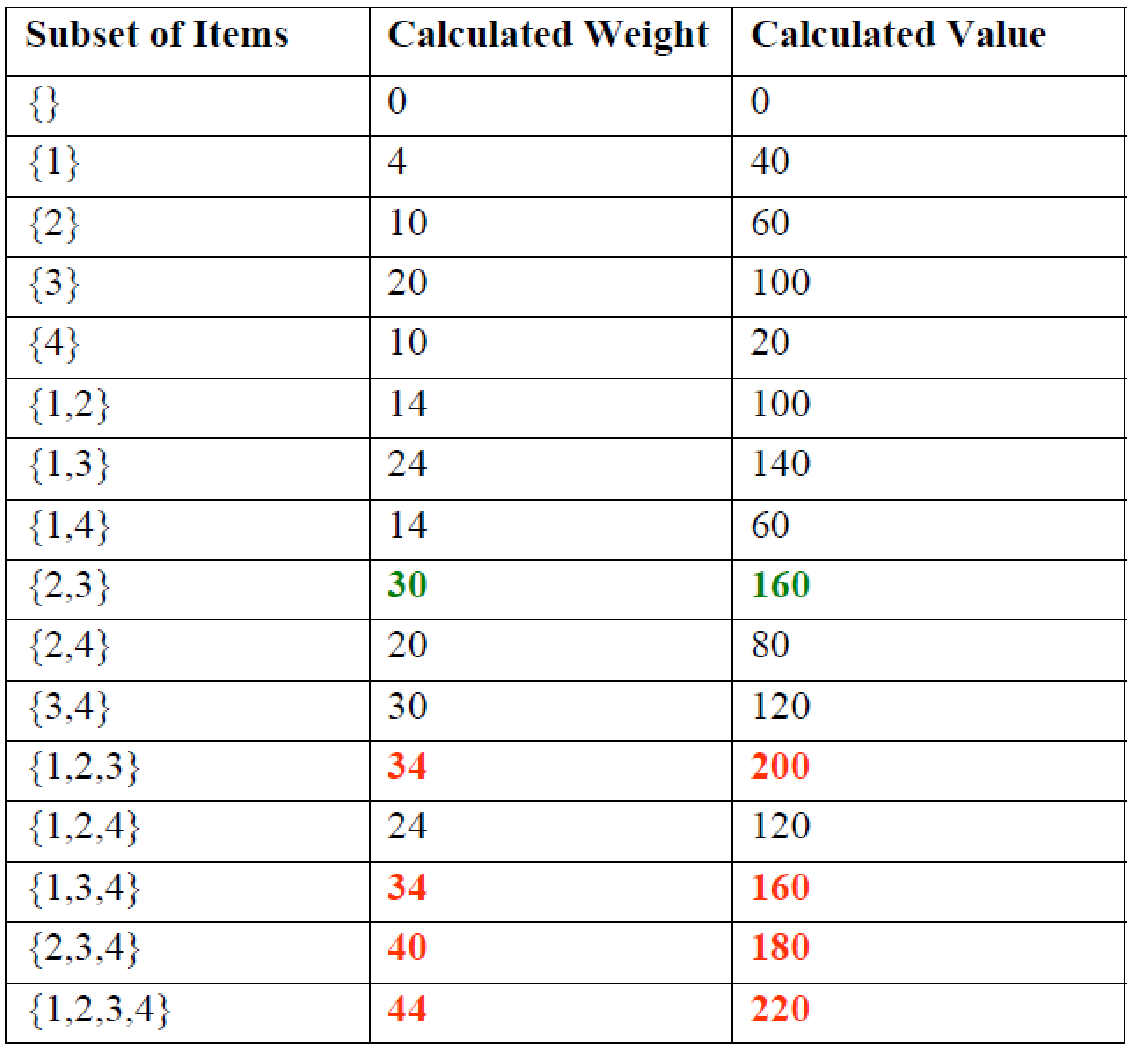
Knapsack Algorithm: Backtracking

```
BackTrack(i, r)          \\ BackTrack(1, C)
b ← 0
{try each kind of item in tern}
for k ← i to n
        do
        if w(k) ≤ r then
                b ← max (b, v[k] + BackTrack(k, r - w[k]))
return b
```

Question: Using Brute Force Method to find an Optimal Solution for the 0-1 Knapsack problem, knapsack capacity W = 32

Item	**Weight**	**Value**	**knapsack capacity W = 32**
1	**4**	**40**	
2	**10**	**60**	
3	**20**	**100**	
4	**10**	**20**	

In Brute force Method to find an optimal solution for the 0-1 Knapsack problem, we find all the subsets of the set of items and calculate the total weight and total value for each. We exclude those subsets for which calculated weight is more than the capacity of knapsack.



Subset of Items	Calculated Weight	Calculated Value
{}	0	0
{1}	4	40
{2}	10	60
{3}	20	100
{4}	10	20
{1,2}	14	100
{1,3}	24	140
{1,4}	14	60
{2,3}	**30**	**160**
{2,4}	20	80
{3,4}	30	120
{1,2,3}	**34**	**200**
{1,2,4}	24	120
{1,3,4}	**34**	**160**
{2,3,4}	**40**	**180**
{1,2,3,4}	**44**	**220**

Here, the subset {2,3} gives maximum value i.e. 160 with total weight of 30 (which is less than the capacity of knapsack i.e. 32).

Hence, choosing items 2 and 3 gives us the maximum value, with weights within the capacity of the knapsack.

Question: Below is a set of 9 points in 2-D. Compute the set of maximal points using Brute Force Approach.

(2, 6), (4, 16), (8, 8), (8, 20), (10, 4), (14, 12), (16, 16), (18, 2), (22, 8).

Solution:

We have to compute the set of maximal points from following points, using Brute Force Approach: **(2, 6), (4, 16), (8, 8), (8, 20), (10, 4), (14, 12), (16, 16), (18, 2), (22, 8)**

In maximal points using Brute Force approach, we have to compare each point with each of the other points. And in each comparison we need to compare both the x and y coordinates of the two points. Hence the procedure for computing maximal points will be as follows:

```
MAXIMAL-POINTS (Point P[1. . . m])
    A = ∅;
    for i ←1 to m          \\ m used for number of points
    do maximal ← true
        for j ← 1 to m
        do
          if (i ≠ j)&((P[i].x ≤ P[j].x)&(P[i].y ≤ P[j].y))
              then maximal ← false;
                break;
        if maximal
            then A = A ∪ P[i]
```

Using the above procedure the computed maximal points are:
(8, 20), (16, 16) and (22, 8).

Procedure:

Points that dominate (2,6) = (4,16), (8,8), (8,20), (14,12), (16,16), (22,8)

Points that dominate (4,16) = (8,20), (16,16)

Points that dominate (8,8) = (8,20), (14,12), (16,16), (22,8)

Points that dominate (8,20) = No point dominate **(8,20)** So it is maximal

Points that dominate (10,4) = (14,12), (16,16), (22,8)

Points that dominate (14,12) = (16,16)

Points that dominate (16,16) = No point dominate **(16,16)** So it is maximal

Points that dominate (18,2) = **(22,8)**

Set of maximal points = (8,20) , (16,16), (22,8)

Question: We have to compute the set of maximal points from following points, using Brute Force Approach:

(2, 8), (6, 16), (8, 8), (10, 20), (12, 6), (16, 14), (18, 18), (20, 2), (24, 8)

Solution:

Points that dominate (2,8) = (6,16), (8,8), (10,20), (16,14), (18,18), (24,8)

Points that dominate (6,16) = (10, 20), (18, 18)

Points that dominate (8,8) = (10, 20), (16, 14), (18, 18), (24, 8)

Points that dominate (10,20) = No point dominate **(10,20)** So it is maximal

Points that dominate (12,6) = (16, 14), (18, 18), (24, 8)

Points that dominate (16,14) = (18, 18)

Points that dominate (18,18) = No point dominate **(18,18)** So it is maximal

Points that dominate (20,2) = (24, 8)

Set of maximal points = (10,20), (18,18), (24,8)

Algorithm

```
MAXIMAL-POINTS (Point P[1. . . m])
    A = $\emptyset$;
    for i $\leftarrow$1 to m        \\ m used for number of points
    do maximal $\leftarrow$ true
        for j $\leftarrow$ 1 to m
        do
            if (i $\neq$ j)&((P[i].x $\leq$ P[j].x)&(P[i].y $\leq$ P[j].y))
                then maximal $\leftarrow$ false;
                    break;
        if maximal
            then A = A $\cup$ P[i]
```

Question: Why Brute Force Approach is not Economical?

This is related to a famous function in combinatorics called the Catalan numbers. Catalan numbers are related with the number of different binary trees on n nodes.

$$P(n) \in \frac{4^n}{n^{3/2}}$$

The dominating term is the exponential 4^n thus P(n) will grow large very quickly and hence this approach is not economical.

Dynamic Programming Formulation

- Let $A_{i..j} = A_i \cdot A_{i+1} \ldots A_j$
- Order of $A_i = p_{i-1} \times p_i$ and order of $A_j = p_{j-1} \times p_j$
- Order of $A_{i..j}$ = rows in A_i x columns in $A_j = p_{i-1} \times p_j$
- At the highest level of parenthesisation,
  $A_{i..j} = A_{i..k} \times A_{k+1..j} \qquad i \le k < j$
- Let m[i, j] = minimum number of multiplications needed to compute $A_{i..j}$, for $1 \le i \le j$ n
- Objective function = finding minimum number of multiplications needed to compute $A_{1..n}$ i.e. to compute m[1, n]
- $A_{i..j} = (A_i.\ A_{i+1} \ldots A_k).\ (A_{k+1}.\ A_{k+2} \ldots A_j) = A_{i..k} \times A_{k+1..j} \qquad i \le k < j$
- Order of $A_{i..k} = p_{i-1} \times p_k$, and order of $A_{k+1..j} = p_k \times p_j$,
- m[i, k] = minimum number of multiplications needed to compute $A_{i..k}$
- m[k+1, j] = minimum number of multiplications needed to compute $A_{k+1..j}$

Mathematical Model

$$m[i,j] = 0$$

$$m[i,j] = \min_{i \le k < j} \left(m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \right)$$

Cost Comparison Brute Force Dynamic Programming

A simple inspection of the nested loop structure yields a running time of $O(n^3)$ for the algorithm. The loops are nested three deep, and each loop index (l , i, and k) takes on at most
n-1 values.

Brute Force Approach: P(n) = C(n - 1) C(n) Є $\frac{4^n}{n^{\frac{3}{2}}}$

Generalization: Sequence of Objects

Although this algorithm applies well to the problem of matrix chain multiplication. Many researchers have noted that it generalizes well to solving a more abstract problem

- Given a linear sequence of objects
- An associative binary operation on those objects hold
- The objective to find a way to compute the cost of performing that operation on any two given objects
- Finally computing the minimum cost for grouping these objects to apply the operation over the entire sequence.

It is obvious that this problem can be solved using chain matrix multiplication, because there is a one to one correspondence between both problems.

Generalization: String Concatenation

One common special case of chain matrix multiplication problem is string concatenation.

- For example, we are given a list of strings.
  - The cost of concatenating two strings of length m and n is for example O(m + n)
  - Since we need O(m) time to find the end of the first string and O(n) time to copy the second string onto the end of it.
  - Using this cost function, we can write a dynamic programming algorithm to fined the fastest way to concatenate a sequence of strings
  - It is possible to concatenate all in time proportional to sum of their lengths, but here we are interested to link this problem with chain matrix multiplication.

Generalization: Parallel Processors

- Another generalization is to solve the problem when many parallel processors are available.
- In this case, instead of adding the costs of computing each subsequence, we just take the maximum, because we can do them both simultaneously.
- This can drastically affect both the minimum cost and the final optimal grouping
- But of course more balanced groupings that keep all the processors busy is more favorable solution
- There exists some more sophisticated approaches to solve this problem

Question: Use Dynamic Programming to find an Optimal Solution for the 0-1 Knapsack problem.

Item	Weight	Value
1	1	1
2	2	6
3	5	18
4	6	22
5	7	28

Knapsack Capacity W = 11

Use the given **Knapsack capacity** to calculate

Solution:

We need to calculate V[n, W], where n is the number of items and W is the capacity of the knapsack.

Base Case:

$V[0, w] = 0, 0 \leq w \leq W$ No items are available

$V[0, w] = -\infty, w < 0$ Invalid

$V[i, 0] = 0, 0 \leq i \leq n$ No capacity available

Recursive Model:

$V[i, w] = \max (V[i-1, w], vi + V[i-1, w - wi])$

for $1 \leq i \leq n, 0 \leq w \leq W$

Now we have to calculate two tables V[5,11] and Keep[5,11], which is as follows:

V[i, w]	**W = 0**	1	2	3	4	5	6	7	8	9	10	11
i = 0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	1
2	0	1	6	7	7	7	7	7	7	7	7	7
3	0	1	6	7	7	18	19	24	25	25	25	25
4	0	1	6	7	7	18	22	24	28	29	29	40
5	0	1	6	7	7	18	22	28	29	34	35	40

To see how the values of tables V[i, w] and Keep[i, w] are calculated, consider the following two example values:

$$\begin{aligned} V[4, 8] &= \max (V[3, 8], V_4 + V[3, 2]) \\ &= \max (25, 22 + 6) \\ &= \max (25, 28) \\ &= 28 \end{aligned}$$

since 4th item is used here, Keep[4, 8] = 1

$$\begin{aligned} V[5, 11] &= \max (V[4, 11], V_5 + V[4, 4]) \\ &= \max (40, 28 + 7) \\ &= \max (40, 35) \\ &= 40 \end{aligned}$$

since 5th item is not used here, Keep[5, 11] = 0

Keep[i, w]	**W = 1**	2	3	4	5	6	7	8	9	10	11
i = 1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	1	1	1	1	1	1	1	1	1
3	0	0	0	0	1	1	1	1	1	1	1
4	0	0	0	0	0	1	0	1	1	1	1
5	0	0	0	0	0	0	1	1	1	1	0

Hence, items no. 3 and 4 gives us the maximum value (i.e. 40).

Question: Use Dynamic Programming to find an optimal solution for the 0-1 Knapsack problem. [Capacity W = 12]

Item	Weight	Value
1	4	40
2	3	30
3	5	40
4	4	50
5	5	70

Solution:

We need to calculate V[n, W], where n is the number of items and W is the capacity of the knapsack.

Base Case:

$V[0, w] = 0, 0 \leq w \leq W$, no items are available

$V[0, w] = -\infty, w < 0$, invalid

$V[i, 0] = 0, 0 \leq i \leq n$, no capacity available

Recursive Model:

$V[i, w] = \max (V[i-1, w], vi + V[i-1, w - w_i])$

for $1 \leq i \leq n, 0 \leq w \leq W$

Now we have to calculate two tables V[5,12] and Keep[5,12], which is as follows:

V[i, w]	**W = 0**	1	2	3	4	5	6	7	8	9	10	11	12
i = 0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	40	40	40	40	40	40	40	40	40
2	0	0	0	30	40	40	40	70	70	70	70	70	70
3	0	0	0	30	40	40	40	70	70	80	80	80	110
4	0	0	0	30	50	50	50	80	90	90	90	120	120
5	0	0	0	30	50	70	70	80	100	120	120	120	150

To see how the values of tables V[i, w] and Keep[i, w] are calculated, consider the following two example values:

V[5, 12] = max (V[4, 12], V_5 + V[4, 7])

= max (120, 70 + 80)

= max (120, 150

= 150

since 5th item is used here, Keep[5, 12] = 1

V[4, 7] = max (V[3, 7], V_4 + V[3, 3])

= max (70, 50 + 30)

= max (70, 80)

= 80

since 4th item is used here, Keep[4, 7] = 1

V[3, 3] = max (V[2, 3], V_3 + V[3, -2])

= max (70, 50 + 0)

= max (70, 50)

= 70

since 3th item is not used here, Keep[3, 3] = 0

V[2, 3] = max (V[1, 3], V_2 + V[1, 0])

= max (0, 30 + 0)

= max (0, 30)

= 30

since 2th item is used here, Keep[2, 3] = 1

Keep[i, w]	**W = 1**	2	3	4	5	6	7	8	9	10	11	12
i = 1	0	0	0	1	1	1	1	1	1	1	1	1
2	0	0	1	0	0	0	1	1	1	1	1	1
3	0	0	0	0	0	1	0	0	1	1	1	1
4	0	0	0	1	1	1	1	1	1	1	1	1
5	0	0	0	0	1	1	0	1	1	1	1	1

Hence, items no. 2, 4 and 5 gives us the maximum value (i.e. 150).

Question: Longest Common Subsequence?

Problem Statement:

In the longest-common-subsequence (LCS) problem, we are given two sequences

$X = <x_1, x_2, \ldots, x_m>$ and $Y = <y_1, y_2, \ldots, y_n>$

And our objective is to find a maximum-length common subsequence of X and Y.

Note:

This LCS problem can be solved using brute force approach as well but using dynamic programming it will be solved more efficiently.

Longest Common Subsequence: Brute Force Approach

- First we enumerate all the subsequences of $X = <x_1, x_2, \ldots, x_m>$.
- There will be 2^m such subsequences.
- Then we check if a subsequence of X is also a subsequence of Y.
- In this way, we can compute all the common subsequences of X and Y.
- Certainly, this approach requires exponential time, making it impractical for long sequences.

Note:

Because this problem has an optimal sub-structure property, and hence can be solved using approach of dynamic programming

Longest Common Subsequence: Dynamic Programming Solution

Towards Optimal Substructure of LCS: Prefixes

- As we shall see, the natural classes of sub-problems correspond to pairs of "prefixes" of the two input sequences.
- To be precise, given a sequence $X = <x_1, x_2, ..., x_m>$, we define the ith prefix of X, for $i = 0, 1, ..., m$, as $X_i = <x_1, x_2, ..., x_i>$.

Examples,

If X = <A, B, C, B, D, A, B> then

X_4 = <A, B, C, B> and

X_0 is the empty sequence = < >

If $X = (x_1, x_2, \ldots, x_m)$, and $Y = (y_1, y_2, \ldots, y_n)$ be sequences and let us suppose that $Z = (z_1, z_2, \ldots, z_k)$ be a longest common sub-sequence of X and Y

Let, $X_i = (x_1, x_2, \ldots, x_i)$, $Y_j = (y_1, y_2, \ldots, y_j)$ and $Z_l = (z_1, z_2, \ldots, z_l)$ are prefixes of X, Y and Z respectively.

1. if $x_m = y_n$, then $z_k = x_m$ and Z_{k-1} is LCS of X_{m-1}, Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is LCS of X_{m-1} and Y
3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is LCS of X and Y_{n-1}

Finding the LCS Length

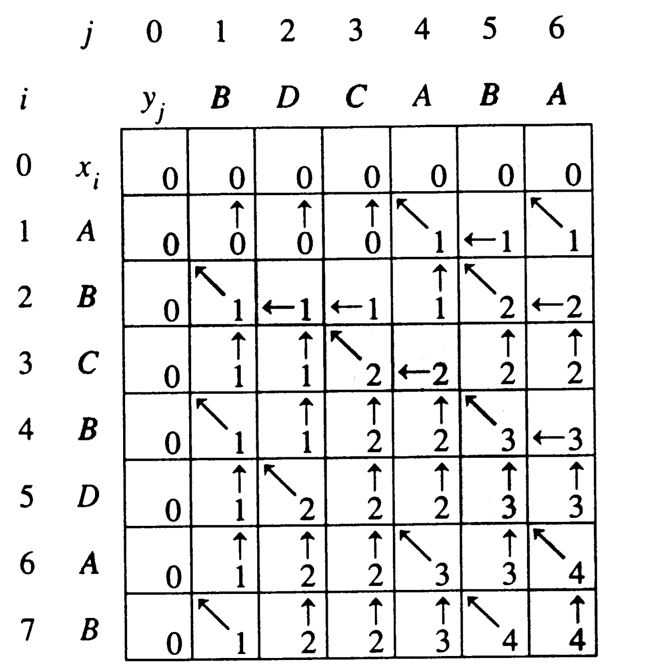
```
LCS-LENGTH(X, Y)
1   m ← length[X]
2   n ← length[Y]
3   for i ← 1 to m
4        do c[i, 0] ← 0
5   for j ← 0 to n
6        do c[0, j] ← 0
7   for i ← 1 to m
8        do for j ← 1 to n
9               do if x_i = y_j
10                    then c[i, j] ← c[i − 1, j − 1] + 1
11                         b[i, j] ← "↖"
12                    else if c[i − 1, j] ≥ c[i, j − 1]
13                            then c[i, j] ← c[i − 1, j]
14                                 b[i, j] ← "↑"
15                            else c[i, j] ← c[i, j − 1]
16                                 b[i, j] ← "←"
17   return c and b
```

- If one sequence is empty the LCS has length 0
- The table b[1..m,1..n] points to the table entry corresponding to the optimal sub-problem solution when computing c[i,j]
- Lines 7 through 16 compute the tables b and c in row major order; the structure matches the three cases outlined ($x_m = y_n$ and two cases for $x_m \neq y_n$)

The figure on the next slide (sample solution) shows how the calculations are performed

Since the table c is (m+1) x (n+1) and each entry takes O(1) to compute, the complexity of the algorithm is Θ(mn)

Longest Common Sequence: A Sample Solution

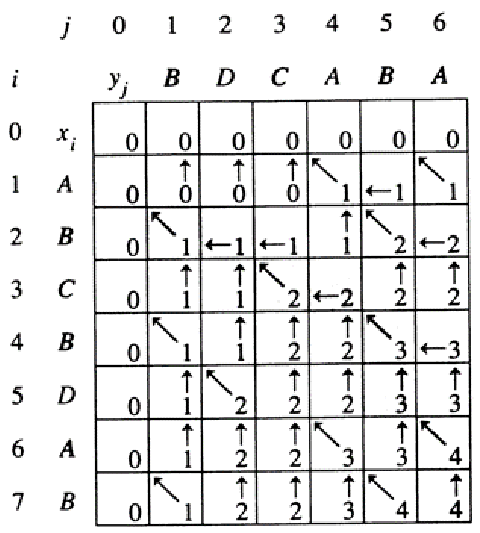


	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

- The answer is 4 from the c[m,n] entry
- If the letters match, the arrow points ↖ and the entry is one greater
- If the letters do not match the value is the maximum of left and up and the arrow points to the larger of left or up (and up if equal)

Constructing Longest Common Sequence (LCS)

Constructing an LCS



	j	0	1	2	3	4	5	6
i		y_j	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑0	↑0	↑0	↖1	←1	↖1
2	B	0	↖1	←1	←1	↑1	↖2	←2
3	C	0	↑1	↑1	↖2	←2	↑2	↑2
4	B	0	↖1	↑1	↑2	↑2	↖3	←3
5	D	0	↑1	↖2	↑2	↑2	↑3	↑3
6	A	0	↑1	↑2	↑2	↖3	↑3	↖4
7	B	0	↖1	↑2	↑2	↑3	↖4	↑4

```
PRINT-LCS(b, X, i, j)
1  if $i = 0$ or $j = 0$
2      then return
3  if $b[i, j] =$ "↖"
4      then PRINT-LCS(b, X, $i - 1$, $j - 1$)
5           print $x_i$
6  elseif $b[i, j] =$ "↑"
7      then PRINT-LCS(b, X, $i - 1$, j)
8  else PRINT-LCS(b, X, i, $j - 1$)
```

- Intuitively you start at c[m,n] and follow the arrows; you print the character whenever the arrow is ↖
- This would print an LCS in reverse order, so the actual code is written recursively with the printing done on exit from recursion; this will print the LCS from left to right

The complexity of printing an LCS is $\Theta(m + n)$

Question: Using Dynamic Programming, determine a Longest Common Subsequence of <1, 0, 0, 1, 0, 1, 0, 1> and <0, 1, 0, 1, 1, 0, 1, 1, 0>.

Solution:

We know,

$$c(i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ c(i-1, j-1) + 1 & \text{if } i, j > 0 \; \& \; x_i = y_j \\ \max(c(i-1, j),\ c(i, j-1)) & \text{if } i, j > 0 \; \& \; x_i \neq y_j \end{cases}$$

where c(i, j) is the length of an LCS of the sequences X_i and Y_j.

Let Xi = <1, 0, 0, 1, 0, 1, 0, 1> and Yj = <0, 1, 0, 1, 1, 0, 1, 1, 0>. Now we calculate the table c[i, j] and b[i, j] as follows:

$x_i \backslash y_j$		**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
	0	0	0	0	0	0	0	0	0	0
1	0	↑0	↖1	←1	↖1	↖1	←1	↖1	↖1	←1
0	0	↖1	↑1	↖2	←2	←2	↖2	←2	←2	↖2
0	0	↖1	↑1	↖2	↑2	↑2	↖3	←3	←3	↖3
1	0	↑1	↖2	↑2	↖3	↖3	↑3	↖4	↖4	←4
0	0	↖1	↑2	↖3	↑3	↑3	↖4	↑4	↑4	↖5
1	0	↑1	↖2	↑3	↖4	↖4	↑4	↖5	↖5	↑5
0	0	↖1	↑2	↖3	↑4	↑4	↖5	↑5	↑5	↖6
1	0	↑1	↖2	↑3	↖4	↖5	↑5	↖6	↖6	↑6

Hence, an LCS of sequences <1, 0, 0, 1, 0, 1, 0, 1> and <0, 1, 0, 1, 1, 0, 1, 1, 0> is **<1 0 0 1 1 0> of length 6**.

Note: There can be many Longest Common Subsequences of length 6 (e.g. as shown in the table).

Question: Use Dynamic programming to determine a Longest Common Subsequence of <1,0,0,1,0,1,0,1> and <0,1,0,1,1,0,1,1,0>

Solution:

We know,

$$c(i,j)=\begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c(i-1,j-1)+1 & \text{if } i,j>0 \,\&\, x_i=y_j \\ \max(c(i-1,j),\ c(i,j-1)) & \text{if } i,j>0 \,\&\, x_i\neq y_j \end{cases}$$

where c(i, j) is the length of an LCS of the sequences X_i and Y_j.

Let Xi = <1, 0, 0, 1, 0, 1, 0, 1> and Yj = <0, 1, 0, 1, 1, 0, 1, 1, 0>. Now we calculate the table c[i, j] and b[i, j] as follows:

xi\yj		**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
	0	0	0	0	0	0	0	0	0	0
1	0	↑0	↖1	←1	↖1	↖1	←1	↖1	↖1	←1
0	0	↖1	↑1	↖2	←2	←2	↖2	←2	←2	↖2
0	0	↖1	↑1	↖2	↑2	↑2	↖3	←3	←3	↖3
1	0	↑1	↖2	↑2	↖3	↖3	↑3	↖4	↖4	←4
0	0	↖1	↑2	↖3	↑3	↑3	↖4	↑4	↑4	↖5
1	0	↑1	↖2	↑3	↖4	↖4	↑4	↖5	↖5	↑5
0	0	↖1	↑2	↖3	↑4	↑4	↖5	↑5	↑5	↖6
1	0	↑1	↖2	↑3	↖4	↖5	↑5	↖6	↖6	↑6

Hence, an LCS of sequences <1, 0, 0, 1, 0, 1, 0, 1> and <0, 1, 0, 1, 1, 0, 1, 1, 0> is **<1 0 0 1 1 0> of length 6**.

Note: There can be many Longest Common Subsequences of length 6 (e.g. as shown in the table).

Question: If X = <A, B, D ,A ,D ,B ,A ,B>, and Y = <D, A ,B ,A ,B ,D ,A ,B , A > are two sequences then compute a maximum-length common subsequence of X and Y.

Solution:

We know,

$$c(i,j)=\begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c(i-1,j-1)+1 & \text{if } i,j>0 \,\&\, x_i=y_j \\ \max(c(i-1,j),\ c(i,j-1)) & \text{if } i,j>0 \,\&\, x_i\neq y_j \end{cases}$$

Where c(i, j) is the length of an LCS of the sequences X_i and Y_j.

Let X = <A, B, D ,A ,D ,B ,A ,B> and Y = <D, A ,B ,A ,B ,D ,A ,B , A >. Now we calculate the table c[i, j] and b[i, j] as follows:

$x_i \backslash y_j$		**D**	**A**	**B**	**A**	**B**	**D**	**A**	**B**	**A**
	0	0	0	0	0	0	0	0	0	0
A	0	↑0	↖1	←1	↖1	←1	←1	↖1	←1	↖1
B	0	↑1	↑1	↖2	←2	↖2	←2	←2	↖2	←2
D	0	↖1	←1	←2	←2	↑2	↖3	←3	←3	←3
A	0	↑1	↖2	←2	↖3	←3	↑3	↖4	←4	↖4
D	0	↖1	↑2	←2	←2	←2	↖4	↑4	←4	←5
B	0	↑1	↑2	↖3	←3	↖3	←3	↑4	↖5	←5
A	0	↑1	↖2	↑3	↖4	←4	←4	↖4	↑5	↖6
B	0	↑1	↑2	↖3	↑4	↖5	←5	←5	↖5	↑6

Hence, an LCS of sequences <A, B, D ,A ,D ,B ,A ,B> and Y = <D, A ,B ,A ,B ,D ,A ,B , A > is **<A B D A B A> of length 6**

Note: There can be many Longest Common Subsequences of length 6 (e.g. as shown in the table).

Question: Find the time complexity of each line of following algorithm.

```
1. Procedure sort(A[1…..n])
2. for i:=1 to n-1 do
3. for j:= to n-I do
4. if A[j] > A [j+1] then
5. Swap A[j] with A[j+1]
```

Solution:

The above algorithm is a bubble sort algorithm; step by step analysis of algorithm is as follows

1- Consist of two nested loop

2- Compute in worst case for an array of size "n"

3- Outer loop is executed *n-1* times, suppose the cost of checking the loop condition and decrementing "i" is c1.

4- 1st passage through the inner loop, *n-1* comparisons and *n-1* swaps (n - 1)st and passage through the inner loop consist of one comparison and one swap.

5- Gathering information of inner loop we get c ((n-1) + (n-2) + ... + 1)

Where "c" is the time required to do one comparison, one swap, check the inner loop condition and increment j

6- We also spend constant time k declaring "i", "j" and initializing "i", "j".

So finally we get

$$c\,((n-1) + (n-2) + \dots + 1) + k + c1(n-1)$$

But

$$(n-1) + (n-2) + \dots + 1 = n(n-1)/2$$

So our function equals

$$c(n(n-1)/2 + k + c1(n-1) = 1/2c\,(n^2 - n) + c1(n-1) + k$$

Which clearly shows the $O(n^2)$ complexity of the algorithm

Question: Give a comparison of Divide & Conquer Approach and Dynamic Programming. Give the reasoning that why the Optimal Weight Convex Polygon Problem cannot be solved using Divide and Conquer Approach.

Solution:

Comparison of Divide and Conquer approach and Dynamic Programming:

Divide and Conquer Approach	Dynamic Programming
Used when sub-problems are independent of each other. So, pick partition that makes algorithm most efficient & simply combine solutions to solve entire problem.	Used when sub-problems are dependent on each other. We don't know where to partition the problem.
Used to find a solution of a problem	Used to find an Optimal Solution of the problem.
Combines the sub-problem solutions on the basis of some predefined functions.	Combine the sub-problem solutions on the basis of optimal value.
Divide-&-conquer is best suited for the case when no "overlapping sub-problems" are encountered.	In dynamic programming algorithms, we typically solve each sub-problem only once and store their solutions. But this is at the cost of space.

Reasoning:

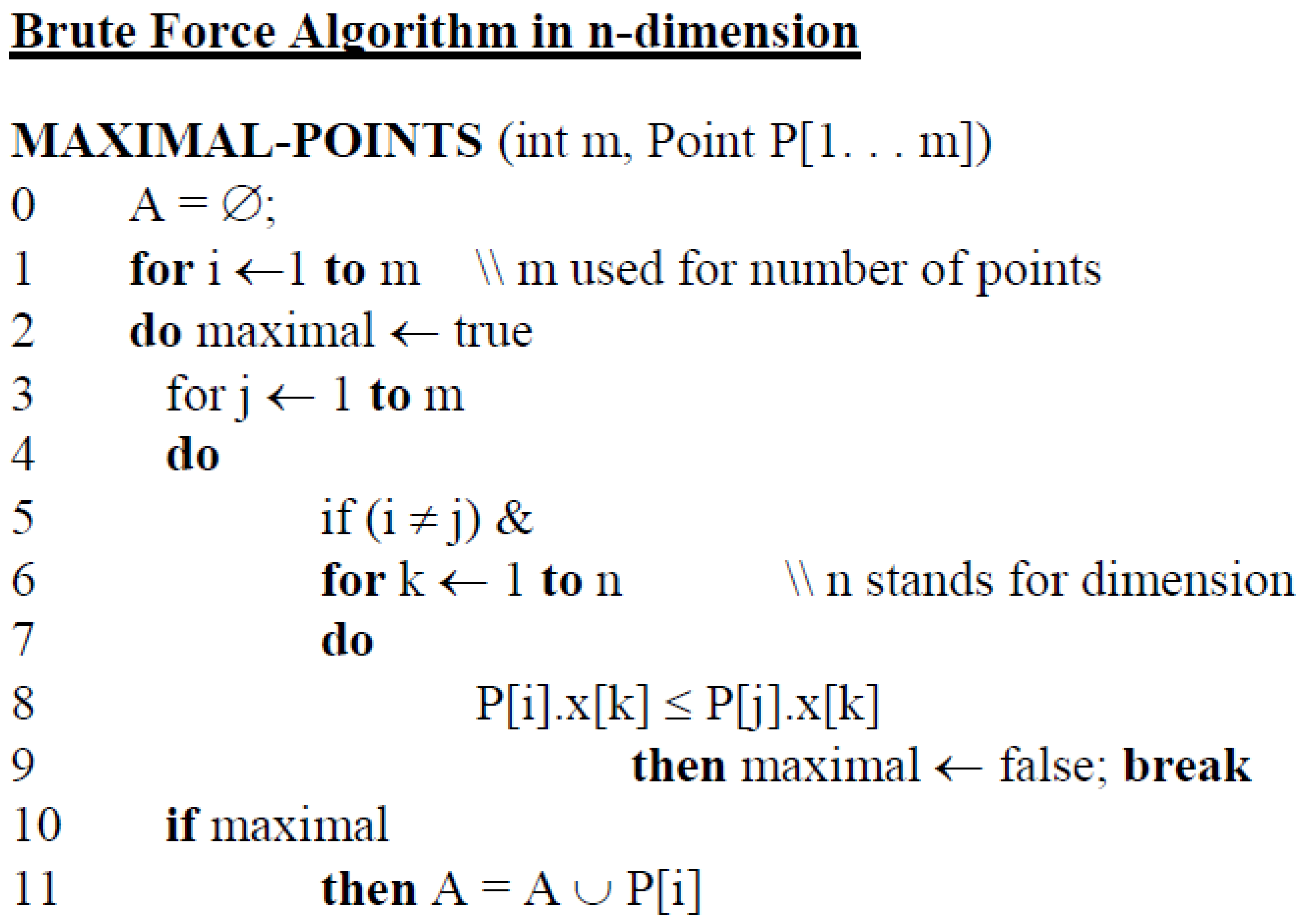
A simple polygon is convex if given any two points on its boundary or in its interior all points on the line segment drawn between them are contained in the polygon's boundary or interior.

Optimal Weight Convex polygon problem cannot be solved using divide and conquer approach because:

- Its sub problems are not independent from each other.
- We need an optimal solution of the problem, which may not be found using divide and conquer approach.

Question: Brute Force Algorithm in n-Dimension | n-Line Assembly Language.

Solution: MAXIMAL-POINTS



Brute Force Algorithm in n-dimension

```
MAXIMAL-POINTS (int m, Point P[1. . . m])
0    A = ∅;
1    for i ←1 to m    \\ m used for number of points
2    do maximal ← true
3       for j ← 1 to m
4       do
5             if (i ≠ j) &
6             for k ← 1 to n          \\ n stands for dimension
7             do
8                   P[i].x[k] ≤ P[j].x[k]
9                         then maximal ← false; break
10      if maximal
11            then A = A ∪ P[i]
```

Question: Plane Sweep Algorithm in n-Dimension?

Write pseudo code of Plane Sweep algorithm in n-Dimension?

Solution:

Plane Sweep Algorithm in n-dimension

```
MAXIMAL-PINTS (int m, int n, Point P[1. . . m])
1       sort P in increasing order by first component
2       stack s;
3       for i ←1 to m    \\ m used for number of points
4       do
5         while (s.noEmpty() &
6         for j ← 2 to n \\ n stands for dimension
7         do
8                   s.top().x[j] ≤ P[i].x[j])
9         do s.pop();
10        s.push(P[i]);
11      output the contents of stack s;
```

Optimal Weight Triangulation

Why Polygon Triangulation?

- Finite element method is a technique for solving numerical problems e.g. stress or heat flow simulations of any kind of systems
- It involves dividing a shape into simple elements for example triangles
- Then formulating a set of linear equations describing relations between simulated quantities in each element, and solving these equations.
- The time and accuracy both, in solution, depend on the quality of dividing into triangles
- Generally, it is desired that triangles must be as close to equilateral as possible

Similarity: Optimal Polygon Triangulation, other Problem

- Optimal triangulation problem is very similar to matrix chain multiplication
- It is an excellent approach to make one to one corresponding between two problems and
- Then solving one problem based on the approach already used in the solution of the other problem
- This is what we are going to do in solving an optimal solution of the triangulation problem which is very popular in computational geometry
- Applications of this problem can be observed in many other areas where division of structures is required before performing computation over it.

Basic Concepts

- **Polygon:** A set of finite piecewise-linear, closed curve in a plane is called a polygon
- **Sides:** The pieces of the polygon are called its sides
- **Vertex:** A point joining two consecutive sides is called a vertex
- **Interior:** The set of points in the plane enclosed by a simple polygon forms interior of the polygon
- **Boundary:** The set of point on the polygon forms its boundary
- **Exterior:** The set of points surrounding the polygon form its exterior
- **Simple Polygon:** A polygon is simple if it does not cross itself, i.e., if its sides do not intersect one another except for two consecutive sides sharing a common vertex.
- **Subdivision of Polygon:** A simple polygon subdivides the plane into its interior, its boundary and it's exterior.
- **Convex Polygon:** A simple polygon is convex if given any two points on its boundary or in its interior, all points on the line segment drawn between them are contained in the polygon's boundary or interior.

Proof of Lemmas

Lemma 1: A triangulation of a simple polygon, with n vertices, has n-2 number of triangles.

Proof:

Proof is done using mathematical induction

Basis Step

Suppose that there three vertices, polygon will be a triangle, i.e. there are 3 - 2 = 1 number of triangles. Hence statement is true for n = 3

If there are 4 vertices, polygon will be in fact a rectangle, divide it into two triangles. The result is true. Hence statement is true for n = 4

Inductive Hypothesis

Let us suppose that statement is true for n = k, i.e., if there are k vertices then there are k-2 number of triangles

Claim:

Now we have to prove that if there are k+1 vertices there must be k+1-2 = k-1, number of triangles.

Since for *k* vertices there are k-2 triangles. Insert one more point at boundary of polygon

In fact point will be inserted at boundary of one of the triangles. So the triangle will become rectangle. Divide it into two triangles. It will increase one more triangle in the division.

Hence it becomes; k – 2 + 1 = k - 1, number of triangles.

It proves the claim. Hence by mathematical induction it proves that for n number of vertices there are n - 2 number of triangles.

Lemma 2: A triangulation of a simple polygon, with n vertices, has n-3 chords.

Proof:

If there are three points, it will be triangle. To make a chord in a polygon, it requires at least four points. So we have to give proof for n ≥ 4.

Basis Step

Suppose that there are four number of vertices, in this case polygon will be a rectangle, there must be 4 - 3 = 1 number of chords. Hence statement is true for n = 4
Inductive Hypothesis

Let us suppose that the statement is true for n = k, i.e., if there are k vertices then there are k - 3 number of chords of the polygon.

Claim

Now we have to prove that if there are k+1 vertices there must be k+1-3 = k-2, number of chords.

Since for k vertices there are k-3 chords. Insert one more point at boundary of polygon

In fact point will be inserted at boundary of one of the triangles. So the triangle will become rectangle. Divide it into two triangles. It will increase one more chord in the division. Hence it becomes k – 3 + 1 = k - 2, number of chords.

Proved.

Question: In the optimal weight triangulation problem, prove that for a triangulation of a convex polygon with n vertices, there will be n-1 number of leaf nodes for the resultant binary tree generated from the dual graph.

Solution:

A convex polygon with *n* vertices $<v_0, v_1, v_2, \ldots, v_{n-1}>$ will have *n* no. of sides. We fix the side v_0v_{n-1}. We know, through Lemma 1, a convex polygon with *n* vertices has *n-2* no. of triangles. Considering its dual graph, the root node of the binary tree is the triangle which contains the side v_0v_{n-1}. Each of the other triangles corresponds to an internal node of the binary tree. Each of the sides of the convex polygon except v_0v_{n-1} corresponds to a leaf node of the resultant binary tree. Hence, there is *n-1* no. of leaf nodes of the resultant binary tree.

Now proving the above statement using mathematical induction:

Basis Step:
Suppose there are three vertices $<v_0, v_1, v_2>$, i.e. the polygon is a triangle. Each side except the v_0v_2 (which are v_0v_1 and v_1v_2), corresponds to a leaf node for the resultant binary tree. Also, 3-1 = 2 no. of leaf nodes. Hence, the statement is true for n = 3.
Inductive Hypothesis:
Let us suppose that the statement is true for $n = k$, i.e. if there are *k* vertices of the convex polygon then there will be *k-1* no. of leaf nodes in the resultant binary tree.

Claim:
Now, we have to prove that if there are *k+1* vertices, there will be *k* no. of leaf nodes in the binary tree.

Since for *k* vertices polygon there are *k-1* no. of leaf nodes in the binary tree. We insert one point at boundary of the polygon. This will divide one of the sides into two, resulting total *k+1* no. of sides. Now, excluding v_0v_k, there will be *k* no. of sides, each corresponding to a leaf node of the resultant binary tree. It proves our claim.

Hence, the statement is true.

Question: Prove that for all integers n, if n^2 is even then n is also even?

Proof

- Express the above statement in the form:

  $\forall x \in D, P(x) \Rightarrow Q(x)$
- Suppose that

  $D = Z$,

  $\text{Even}(n, 2) \equiv n^2$ is even

  $\text{Even}(n) \equiv n$ is even
- We have to prove that

  $\forall n \in Z, \text{Even}(n, 2) \Rightarrow \text{Even}(n)$
- Contraposition of the above statement

  $\forall n \in Z, \neg \text{Even}(n) \Rightarrow \neg \text{Even}(n, 2)$ is even
- Now we prove above contrapositive by direct proof
- Suppose that n is an arbitrary element of Z such that, $\neg$ Even(n) (n is not even) i.e., n is odd
- n^2 = n.n = odd. odd = odd
- n^2 is odd

- $\neg$ Even(n, 2) is even
- Hence, $\forall n \in Z, \neg \text{Even}(n) \Rightarrow \neg \text{Even}(n, 2)$ is even
- Therefore, $\forall n \in Z, \text{Even}(n, 2) \Rightarrow \text{Even}(n)$ is even
- Hence $\forall n \in Z$, if n^2 is even then n is even

Question: Fractional Knapsak Psuedo Code?

Developing of algorithm if asked in the question.

Developing Algorithm: Fractional Knapsack

- Pick the item with the maximum value per pound v_i/w_i
- If the supply of that element is exhausted and the thief can carry more then take as much as possible from the item with the next greatest value per pound
- Continue this process till knapsack is filled
- It is good to order items based on their value per pound

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$$

Algorithm: Fractional Knapsack Problem

```
Fractional-Knapsack (W, v[n], w[n])
    1.      While w > 0 and as long as there are items remaining
    2.          pick item with maximum $v_i/w_i$
    3.          $x_i \leftarrow \min(1, w/w_i)$
    4.          remove item i from list
    5.          $w \leftarrow w - x_i w_i$
```

w the amount of space remaining in the knapsack (w = W)

Running time: $\Theta(n)$ if items already ordered; else $\Theta(n\lg n)$

Problem Statement: Chain Matrix Multiplication

Given a chain of [A_1, A_2, . . . , A_n] of n matrices where for i = 1, 2, . . . , n,

Matrix A_i has dimension p_{i-1} x p_i,

Find the order of multiplication which minimizes the number of scalar multiplications.

Note:

Order of A_1 is p_0 x p_1,

Order of A_2 is p_1 x p_2,

Order of A_3 is p_2 x p_3, etc.

Order of A_1 x A_2 x A_3 is p_0 x p_3,

Order of A_1 x A_2 x . . . x A_n is p_0 x p_n

Why Dynamic Programming in this problem?

- Problem is of type optimization
- Sub-problems are dependent
- Optimal structure can be characterized and
- Can be defined recursively
- Solution for base cases exits
- Optimal solution can be constructed
- Hence here is dynamic programming

Dynamic Programming Formulation

- Let $A_{i..j} = A_i \, . \, A_{i+1} \, . \, . \, . \, A_j$
- Order of $A_i = p_{i-1}$ x p_i and order of $A_j = p_{j-1}$ x p_j
- Order of $A_{i..j}$ = rows in A_i x columns in $A_j = p_{i-1} \times p_j$
- At the highest level of parenthesisation,
  $A_{i..j} = A_{i..k} \times A_{k+1..j} \qquad i \le k < j$
- Let m[i, j] = minimum number of multiplications needed to compute $A_{i..j}$, for $1 \le i \le j \le n$
- Objective function = finding minimum number of multiplications needed to compute $A_{1..n}$ i.e. to compute m[1, n]
- $A_{i..j} = (A_i. \, A_{i+1} \ldots\ldots A_k). \, (A_{k+1}. \, A_{k+2} \ldots\ldots A_j) = A_{i..k} \times A_{k+1..j} \qquad i \le k < j$
- Order of $A_{i..k} = p_{i-1}$ x p_k, and order of $A_{k+1..j} = p_k$ x p_j,
- m[i, k] = minimum number of multiplications needed to compute $A_{i..k}$
- m[k+1, j] = minimum number of multiplications needed to compute $A_{k+1..j}$

Mathematical Model

$$m[i, j] = 0$$

$$m[i, j] = \min_{i \le k < j} \left(m[i, k] + m[k+1, j] + p_{i-1} p_k p_j \right)$$

Question: Why we use dynamic programming? Give limitations?

Answer:

- Dynamic programming can be applied to any problem that observes the principle of optimality.
- Generally, it means that partial solutions can be optimally extended with regard to the state after the partial solution instead of the partial solution itself.
- The biggest limitation using dynamic programming is number of partial solutions we must keep track of
- For all examples we have seen, partial solutions can be described by stopping places in the input.
- This is because combinatorial objects e.g. strings, numerical sequences, and polygons etc., all have an implicit order defined upon their elements.
- This order cannot be changed without completely changing the original problem.
- Once order fixed, there are relatively few possible stopping places, and we get an efficient algorithms.

Why Dynamic Programming?

- Dynamic programming, like divide and conquer method, solves problems by combining the solutions to sub-problems.
- Divide and conquer algorithms:
- Partition the problem into independent sub-problem
- Solve the sub-problem recursively and
- Combine their solutions to solve the original problem
- In contrast, dynamic programming is applicable when the sub-problems are not independent.
- Dynamic programming is typically applied to optimization problems.

Question: Write down the Brute Force Chain Matrix Multiplication Algorithm and its complexity.

Generalization of Brute Force Approach

If there is sequence of n matrices, [A1, A2, . . . , An].

A_i has dimension p_{i-1} x p_i, where for i = 1, 2, . . . , n.

Find order of multiplication that minimizes number of scalar multiplications using brute force approach.

Recurrence Relation: After kth matrix, create two sub-lists, one with k and other with n - k matrices i.e. $(A_1 A_2 A_3 A_4 A_5 \ldots A_k)(A_{k+1} A_{k+2} \ldots A_n)$

Let P(n) be the number of different ways of parenthesizing n items

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$$

If n = 2

$P(2) = P(1).P(1) = 1.1 = 1$

If n = 3

$P(3) = P(1).P(2) + P(2).P(1) = 1.1 + 1.1 = 2$

$(A_1 A_2 A_3) = ((A_1 . A_2). A_3)$ OR $(A_1 . (A_2. A_3))$

If n = 4

$P(4) = P(1).P(3) + P(2).P(2) + P(3).P(1) = 1.2 + 1.1 + 2.1 = 5$

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$$

Why Brute Force Approach is not Economical?

This is related to a famous function in combinatorics called the Catalan numbers. Catalan numbers are related with the number of different binary trees on n nodes.

$P(n) \in (4^n/n^{3/2})$

The dominating term is the exponential 4^n thus P(n) will grow large very quickly and hence this approach is not economical.

Question: Algorithm of chain matrix multiplication

Brute Force Approach

- If we wish to multiply two matrices: $A = a[i, j]_{p,\, q}$ and $B = b[i, j]_{q,\, r}$
- Now if C = AB then order of C is p x r.
- Since in each entry c[i, j], there are q number of scalar of multiplications
- Total number of scalar multiplications in computing

  C = Total entries in C x Cost of computing a single entry = p . r . q
- Hence the computational cost of AB = p . q . r will be

In particular, for $1 \le i \le p$ and $1 \le j \le r$

$$C[i, j] = \sum_{k=1}^{q} A[i, k]\, B[k, j]$$

There are p . r total entries in C and each takes O(q) time to compute, thus the total time to multiply these two matrices is dominated by the number of scalar multiplication, which is p . q . r.

Dynamic Programming: Matrix Chain Multiplication

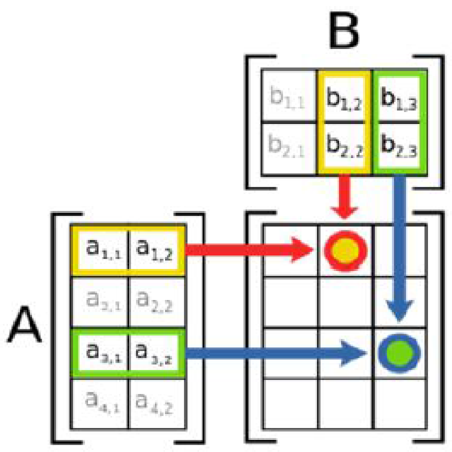
Standard algorithm for multiplying two matrices

If A is a *p x q* and B is a *q x r* matrix, then C = AB is a *p x r* matrix.

$$C[i,j] = \sum_{k=1}^{q} A[i,k]B[k,j]$$

Time complexity: O(pqr)

```
MATRIX-MULTIPLY$(A, B)$
1.  if $A$.columns $\neq$ $B$.rows
2.      error "incompatible dimensions"
3.  else let $C$ be a new $A$.rows $\times$ $B$.columns matrix
4.      for $i \leftarrow 1$ to $A$.rows
5.          for $j \leftarrow 1$ to $B$.columns
6.              $c_{ij} \leftarrow 0$
7.              for $k \leftarrow 1$ to $A$.columns
8.                  $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$
9.      return $C$
```



Procedure that prints an optimal parenthesis / parenthesization of a table computed by MATRIX-CHAIN-ORDER.

- The following procedure prints an optimal parenthesization of $\langle A_i, A_{i+1}, \dots, A_j \rangle$, given the s table computed by MATRIX-CHAIN-ORDER and the indices i and j
- The call PRINT-OPTIMAL-PARENS$(s, 1, n)$ prints an optimal parenthesization of $\langle A_1, A_2, \dots, A_n \rangle$

```
PRINT-OPTIMAL-PARENS$(s, i, j)$
1. if $i = j$
2.     print "$A$"$_i$
3. else print "("
4.     PRINT-OPTIMAL-PARENS$(s, i, s[i, j])$
5.     PRINT-OPTIMAL-PARENS$(s, s[i, j] + 1, j)$
6.     print ")"
```

Question: Write Algorithm for Recursive Matric Chain Multiplication?

Solution:

```
RECURSIVE-MATRIX-CHAIN($p, i, j$)
1  if $i = j$
2     then return 0
3  $m[i, j] \leftarrow \infty$
4  for $k \leftarrow i$ to $j - 1$
5       do $q \leftarrow$ RECURSIVE-MATRIX-CHAIN($p, i, k$)
                    + RECURSIVE-MATRIX-CHAIN($p, k + 1, j$) + $p_{i-1} p_k p_j$
6          if $q < m[i, j]$
7             then $m[i, j] \leftarrow q$
8  return $m[i, j]$
```

Question: Algorithm for Closest Pair in 2D using Brute Force approach?

Solution:

Brute Force Approach: Finding Closest Pair in 2-D

```
ClosestPairBF(P)
        1.  mind ← ∞
        2.  for i ← 1 to n
        3.  do
                4.  for j ← 1 to n
                5.  if i ≠ j
                6.  do
                7.  d ← ((x_i − x_j)^2 + (y_i − y_j)^2)
                8.  if d < minn then
                        8.  mind ← d
                        9.  mini ← i
                        10. minj ← j
        11. return mind, p(mini, minj)
```

Time Complexity:

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} c$$

$$= \sum_{i=1}^{n} cn$$

$$= cn^2$$

$$= \Theta(n^2)$$

Question: Brute Force Improved Version: Finding Closest Pair in 2-D

ClosestPairBF(P)

```
1. mind ← ∞
2. for i ← 1 to n − 1
3. do
       4. for j ← i + 1 to n
       5. do
       6. d ← ((x_i − x_j)^2 + (y_i − y_j)^2)
       7. if d < minn then
              8. mind ← d
              9. mini ← i
              10. minj ← j
11. return mind, p(mini, minj)
```

Time Complexity:

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c$$

$$= \sum_{i=1}^{n-1} c(n-i)$$

$$= c(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i)$$

$$= cn(n-1) - c\frac{(n-1)n}{2}$$

$$= \Theta(n^2)$$

Question: Brute Force Approach: Finding Closest Pair in 3-D

```
ClosestPairBF(P)
    1.  mind ← ∞
    2.  for i ← 1 to n − 1
    3.  do
            4.  for j ← i + 1 to n
            5.  do
            6.  d ← $((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)$
            7.  if d < minn then
                    8.  mind ← d
                    9.  mini ← i
                    10. minj ← j
    11. return mind, p(mini), p(minj)
```

Time Complexity

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c$$

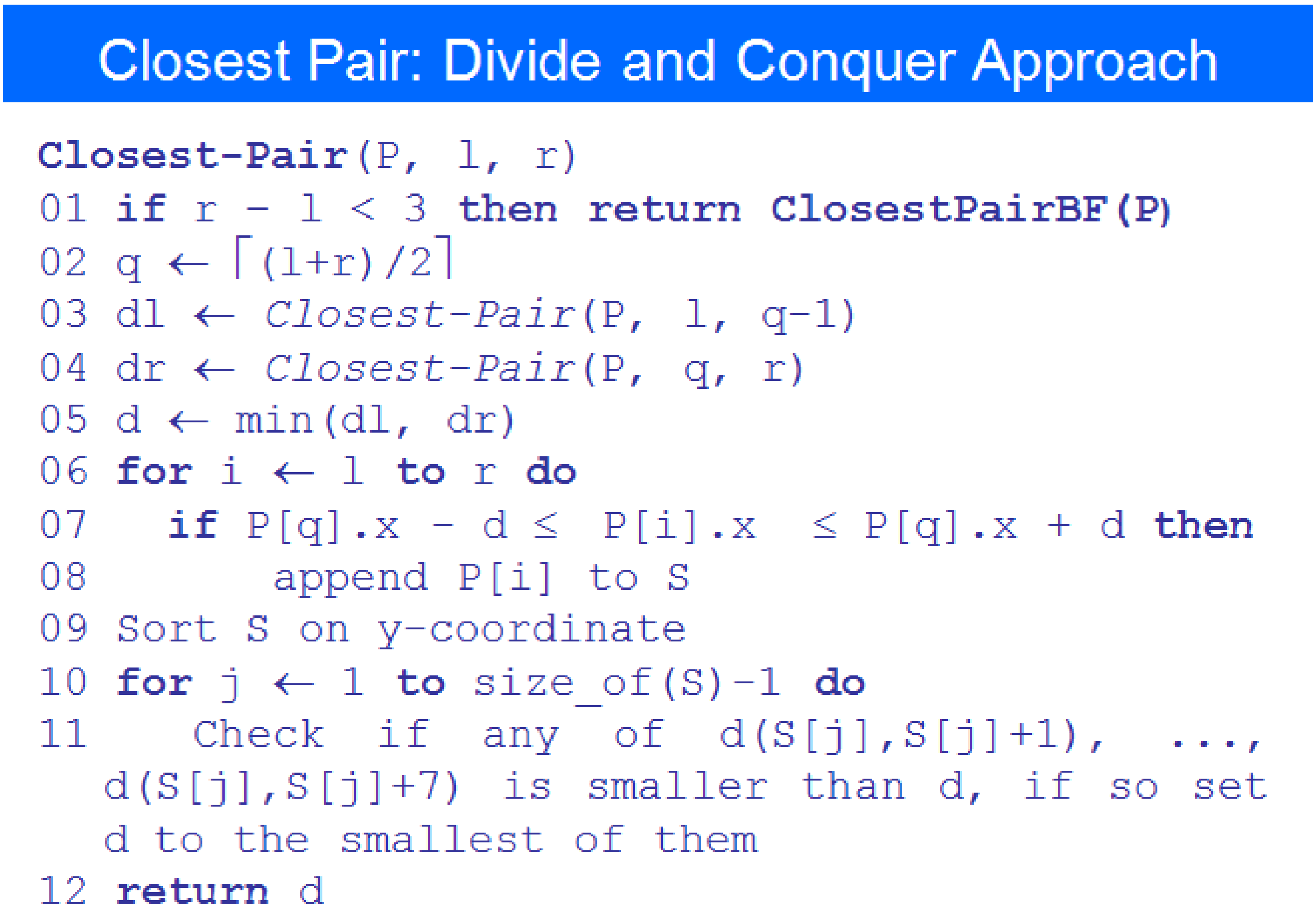
$$= \sum_{i=1}^{n-1} c(n-i)$$

$$= c(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i)$$

$$= \Theta(n^2)$$

Question: Algorithm for Closest Pair in 2-D using Divide and Conquer

Closest Pair in 2-D Algorithm using Divide and Conquer approach.



Closest Pair: Divide and Conquer Approach

```
Closest-Pair(P, l, r)
if r − l < 3 then return ClosestPairBF(P)
q ← ⌈(l+r)/2⌉
dl ← Closest-Pair(P, l, q-1)
dr ← Closest-Pair(P, q, r)
d ← min(dl, dr)
for i ← l to r do
    if P[q].x - d ≤ P[i].x ≤ P[q].x + d then
        append P[i] to S
Sort S on y-coordinate
for j ← 1 to size_of(S)-1 do
    Check if any of d(S[j],S[j]+1), ...,
  d(S[j],S[j]+7) is smaller than d, if so set
  d to the smallest of them
return d
```

Running Time: Running time of a divide-and-conquer algorithm can be described by a recurrence

– Divide = O(1)

– Combine = O(n log n)

– This gives the recurrence given below

– Total running time: O(n $\log_2$ n)

$$T(n) = \begin{cases} n & n \le 3 \\ 2T(\frac{n}{2}) + n\log n & \text{otherwise} \end{cases}$$

Assembly-Line Scheduling Problem

- There are two assembly lines each with n stations
- The j^{th} station on line i is denoted by $S_{i,\,j}$
- The assembly time at that station is $a_{i,j}$.
- An auto enters factory, goes into line i taking time e_i
- After going through the j_{th} station on a line i, the auto goes on to the $(j+1)^{st}$ station on either line
- There is no transfer cost if it stays on the same line
- It takes time $t_{i,j}$ to transfer to other line after station $S_{i,j}$
- After exiting the nth station on a line, it takes time x_i for the completed auto to exit the factory.
- Problem is to determine which stations to choose from lines 1 and 2 to minimize total time through the factory.

Steps in Development of Dynamic Algorithms

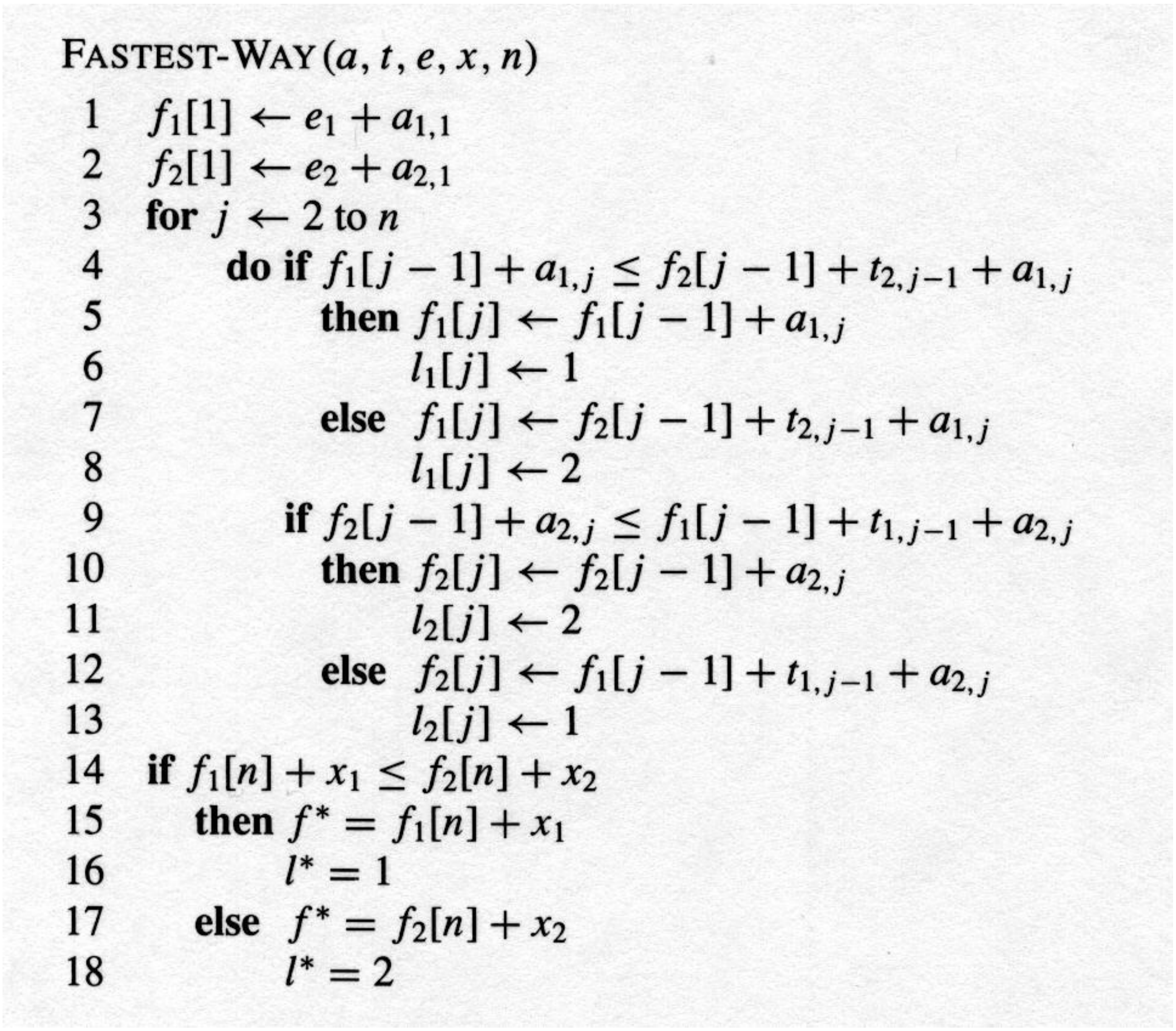
1. Characterize the structure of an optimal solution
2. Recursively define the value of an optimal solution
3. Compute the value of an optimal solution in a bottom-up fashion
4. Construct an optimal solution from computed information

Note: Steps 1-3 form the basis of a dynamic programming solution to a problem. Step 4 can be omitted only if the value of an optimal solution is required.

Question: Pseudo code of n line Assembly?

Write algorithm for n-line Assembly using Dynamic Programming / Algorithm / The Fastest Way

n-Line Assembly: Dynamic Algorithm



```
FASTEST-WAY$(a, t, e, x, n)$
$f_1[1] \leftarrow e_1 + a_{1,1}$
$f_2[1] \leftarrow e_2 + a_{2,1}$
for $j \leftarrow 2$ to $n$
        do if $f_1[j-1] + a_{1,j} \le f_2[j-1] + t_{2,j-1} + a_{1,j}$
              then $f_1[j] \leftarrow f_1[j-1] + a_{1,j}$
                   $l_1[j] \leftarrow 1$
              else $f_1[j] \leftarrow f_2[j-1] + t_{2,j-1} + a_{1,j}$
                   $l_1[j] \leftarrow 2$
           if $f_2[j-1] + a_{2,j} \le f_1[j-1] + t_{1,j-1} + a_{2,j}$
              then $f_2[j] \leftarrow f_2[j-1] + a_{2,j}$
                   $l_2[j] \leftarrow 2$
              else $f_2[j] \leftarrow f_1[j-1] + t_{1,j-1} + a_{2,j}$
                   $l_2[j] \leftarrow 1$
if $f_1[n] + x_1 \le f_2[n] + x_2$
     then $f^* = f_1[n] + x_1$
          $l^* = 1$
     else $f^* = f_2[n] + x_2$
          $l^* = 2$
```

Question: Write algorithm 2-Line Assembly language?

Solution:

The closest pair of points can be computed in O(n^2) time by performing a brute-force search. To do that, one could compute the distances between all the $n(n - 1) / 2$ pairs of points, then pick the pair with the smallest distance, as illustrated below.

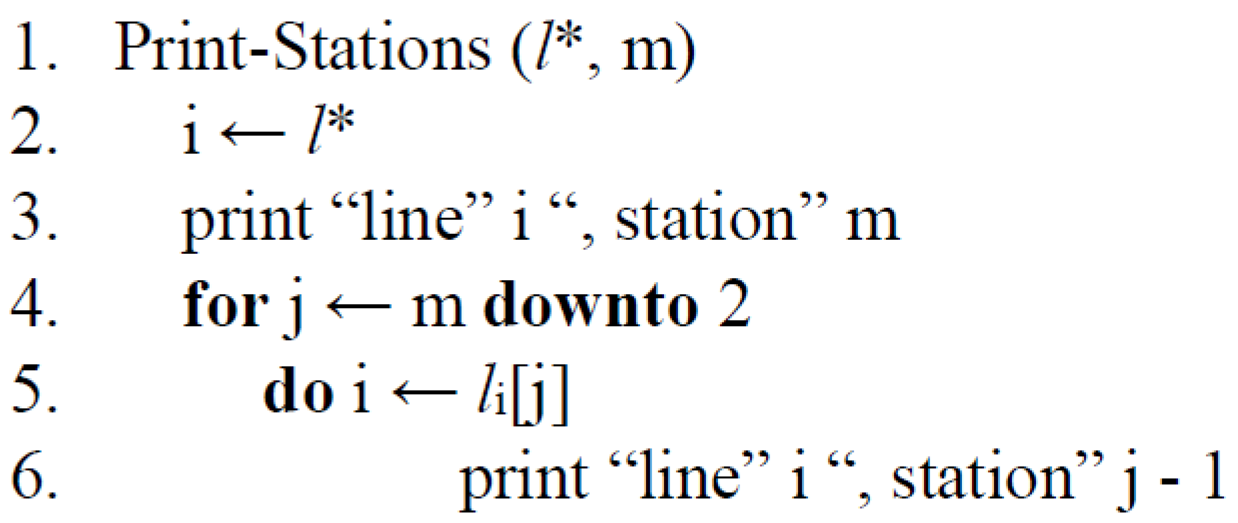
```
minDist = infinity
for i = 1 to length(P) - 1
 for j = i + 1 to length(P)
  let p = P[i], q = P[j]
  if dist(p, q) < minDist:
   minDist = dist(p, q)
   closestPair = (p, q)
return closestPair
```

Question: Print station n - Line assembly

Write pseudo code of n-line assembly: dynamic programming algorithm for print stations?

Write pseudo code of Assembly line scheduling dynamic programming algorithm for print station?

Constructing the Fastest Way: n-Line



```
Print-Stations ($l$*, m)
    i ← $l$*
    print "line" i ", station" m
    for j ← m downto 2
        do i ← $l_i$[j]
            print "line" i ", station" j - 1
```

Generalization: Cyclic Assembly Line Scheduling

Title: Moving policies in cyclic assembly line scheduling

Source: Theoretical Computer Science, Volume 351, Issue (February 2006)

Summary: Assembly line problem occurs in various kinds of production automation. In this paper, originality lies in the automated manufacturing of PC boards.

• In this case, the assembly line has to process number of identical work pieces in a cyclic fashion. In contrast to common variant of assembly line scheduling.

• Each station may process parts of several work-pieces at the same time, and parts of a work-piece may be processed by several stations at the same time.

n-Line Assembly Problem

There are *n* assembly lines each with *m* stations

The j^{th} station on line i is denoted by $S_{i,j}$

The assembly time at that station is $a_{i,j}$.

An auto enters factory, goes into line i taking time e_i

After going through the j^{th} station on a line i, the auto goes on to the $(j+1)^{st}$ station on either line

It takes time $t_{i,j}$ to transfer from line i, station j to line i' and station j+1

After exiting the nth station on a line i, it takes time xi for the completed auto to exit the factory.

Problem is to determine which stations to choose from lines 1 to n to minimize total time through the factory.

n-Line: Brute Force Solution

Total Computational Time = possible ways to enter in stations at level n x one way Cost

Possible ways to enter in stations at level 1 = n^1

Possible ways to enter in stations at level 2 = n^2 . . .

Possible ways to enter in stations at level m = n^m

Total Computational Time = $\Theta(m.m^n)$

Application: Multiprocessor Scheduling

- The assembly line problem is well known in the area of multiprocessor scheduling.
- In this problem, we are given a set of tasks to be executed by a system with n identical processors.
- Each task, T_i, requires a fixed, known time pi to execute.
- Tasks are indivisible, so that at most one processor may be executing a given task at any time
- They are un-interruptible, i.e., once assigned a task, may not leave it until task is complete.
- The precedence ordering restrictions between tasks may be represented by a tree or forest of trees

Recursion Tree Method

• Although substitution method can provide a sufficient proof that a solution to a recurrence is correct, sometimes difficult to give a good guess.

• Drawing out a recursion tree, is a straight forward way to devise a good guess.

• In recursion tree, nodes represent costs of a sub-problems in the set of recursive function invocations.

• We sum costs within each level of the tree to obtain a set of per-level costs.

• And then we sum all per-level costs to determine the total cost of all levels of the recursion.

• Recursion trees are particularly useful when recurrence describes running time of divide and conquer algos.

• Recursion tree is best one used to generate a good guess, which is then verified by substitution method

• When using a recursion tree to generate a good guess, we can often tolerate a small amount of sloppiness since we have to verify it later on.

• If we are careful when drawing out a recursion tree and summing costs, then we can use a recursion tree as a direct proof of a solution to any recurrence of any problem.

• Here, we will use recursion trees directly to prove theorem that forms the basis of the master method.

Symmetry over ϴ [Theeta]

Property: Prove that *f*(*n*) = ϴ (*g*(*n*)) ⇔ *g*(*n*) = ϴ (*f*(*n*))

Proof

Since $f(n) = \Theta(g(n))$ i.e. $f(n) \in \Theta(g(n)) \Rightarrow$

$\exists$ constants $c_1, c_2 > 0$ and $n_0 \in N$ such that

$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \quad \forall\ n \ge n_0$ (1)

(1) $\Rightarrow 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \Rightarrow 0 \le f(n) \le c_2 g(n)$

$\Rightarrow 0 \le (1/c_2) f(n) \le g(n)$ (2)

(1) $\Rightarrow 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \Rightarrow 0 \le c_1 g(n) \le f(n)$

$\Rightarrow 0 \le g(n) \le (1/c_1) f(n)$ (3)

From (2),(3): $0 \le (1/c_2) f(n) \le g(n) \wedge 0 \le g(n) \le (1/c_1) f(n)$

$\Rightarrow \quad 0 \le (1/c_2) f(n) \le g(n) \le (1/c_1) f(n)$

Suppose that $1/c_2 = c_3$, and $1/c_1 = c_4$,

Now the above equation implies that

$0 \le c_3 f(n) \le g(n) \le c_4 f(n),\ \forall\ n \ge n_0$

$\Rightarrow g(n) = \Theta(f(n)),\ \forall\ n \ge n_0$

Hence it proves that,

$$f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$$

Question: $n^2 + 5n + 7 = \Theta(n^2)$ [Theeta]

Proof:

- When $n \geq 1$,

$$n^2 + 5n + 7 \leq n^2 + 5n^2 + 7n^2 \leq 13n^2$$

- When $n \geq 0$,

$$n^2 \leq n^2 + 5n + 7$$

- Thus, when $n \geq 1$

$$1n^2 \leq n^2 + 5n + 7 \leq 13n^2$$

Thus, we have shown that $n^2 + 5n + 7 = \Theta(n^2)$ (by definition of Big-Θ, with $n_0 = 1$, $c_1 = 1$, and $c_2 = 13$.)

Question: $5n^2 - 6n = \Theta(n^2)$ [Theeta]

***Proof*:** To prove $5n^2 - 6n = \Theta(n^2)$, we need to show that $5n^2 - 6n = O(n^2)$ and $5n^2 - 6n = \Omega(n^2)$.
First, show $5n^2 - 6n = O(n^2)$
$5n^2 - 6n <= 5n^2 = O(n^2)$, for all $n >= n_1, n_1 = 1, c_1 = 5$
Now, show $5n^2 - 6n = \Omega(n^2)$
$5n^2 - 6n >= n^2$, for all $n >= n_2, n_2 = 2, c_2 = 1$
Therefore, $5n^2 - 6n = \Theta(n^2)$, for all $n >= n_0, n_0 = 2$

Question: Prove that Prn4ove $5n^2 + 7n \in O(n^2)$ [Big-Oh]

Proof:

$5n^2 + 7n \in O(n^2)$

Big-Oh notation formula is

$0 \leq f(n) \leq cg(n) \qquad \forall\ n \geq n_0$

Suppose that $f(n) = 5n^2 + 7n$ and $g(n) = n^2$

$f(n) \in O(g(n))$?

Now, we have to find values of c_1 and n_0 $\forall\ n \geq n_0$

Computing the values of f(n) and g(n) in the formula

$0 \leq 5n^2 + 7n \leq c.n^2$

Let suppose $c_1 = 6$, then we need to find n_0, for which above is true

Let $n_0 = 1$, then we have

$5(1)^2 + 7(1) \leq 6(1)^2$

$5 + 7 \leq 6$

$12 \leq 6$

Now testing different values of n from 1 to onward

We find for $n_0 = 7$

$5(7)^2 + 7(7) \leq 6(7)^2$

$245 + 49 \leq 294$

$294 \leq 294$

Hence $c_1 = 6\ \forall\ n \geq 7$

Proved that $f(n) \in O(g(n))$, therefore also proved that

$5n^2 + 7n \in O(n^2)$

Question: Prove that Prn4ove $5n^2 + 7n = O(n)^2$ [Big-Oh]

Proof: $5n^2 + 7n = O(n)^2$

Big-Oh notation formula is

$0 \leq f(n) \leq cg(n) \quad \forall\ n \geq n_0$

Suppose that $f(n) = 5n^2 + 7n$ and $g(n) = n^2$

$f(n) \leq O(g(n))$?

Now, we have to find values of c_1 and n_0 $\forall\ n \geq n_0$

Computing the values of f(n) and g(n) in the formula

$0 \leq 5n^2 + 7n \leq c.n^2$

Let suppose $c_1 = 6$, then we need to find n_0, for which above is true

Let $n_0 = 1$, then we have

$5(1)^2 + 7(1) \leq 6(1)^2$

$5 + 7 \leq 6$

$12 \leq 6$

Now testing different values of n from 1 to onward

We find for $n_0 = 7$

$5(7)^2 + 7(7) \leq 6(7)^2$

$245 + 49 \leq 294$

$294 \leq 294$

Hence $c_1 = 6\ \forall\ n \geq 7$

Proved that f(n) = O(g(n)), therefore also proved that

$5n^2 + 7n = O(n)^2$

Question: Show that $\frac{1}{2}n^2 + 3n = \Theta(n^2)$ OR [1/2 n^2 + 3n] [Theeta]

Proof:

- Notice that if $n \geq 1$,

$$\frac{1}{2}n^2 + 3n \leq \frac{1}{2}n^2 + 3n^2 = \frac{7}{2}n^2$$

- Thus,

$$\frac{1}{2}n^2 + 3n = O(n^2)$$

- Also, when $n \geq 0$,

$$\frac{1}{2}n^2 \leq \frac{1}{2}n^2 + 3n$$

- So

$$\frac{1}{2}n^2 + 3n = \Omega(n^2)$$

- Since $\frac{1}{2}n^2 + 3n = O(n^2)$ and $\frac{1}{2}n^2 + 3n = \Omega(n^2)$,

$$\frac{1}{2}n^2 + 3n = \Theta(n^2)$$

Question: Show that $\frac{1}{2}n^2$ - 3n = $\Theta(n^2)$ OR [1/2 n^2 – 3n] [Theeta]

Proof:

- We need to find positive constants c_1, c_2, and n_0 such that

$$0 \le c_1 n^2 \le \frac{1}{2}n^2 - 3\,n \le c_2 n^2 \text{ for all } n \ge n_0$$

- Dividing by n^2, we get

$$0 \le c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

- $c_1 \le \frac{1}{2} - \frac{3}{n}$ holds for $n \ge 10$ and $c_1 = 1/5$
- $\frac{1}{2} - \frac{3}{n} \le c_2$ holds for $n \ge 10$ and $c_2 = 1$.
- Thus, if $c_1 = 1/5$, $c_2 = 1$, and $n_0 = 10$, then for all $n \ge n_0$,

$$0 \le c_1 n^2 \le \frac{1}{2}n^2 - 3\,n \le c_2 n^2 \text{ for all } n \ge n_0.$$

Thus we have shown that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$.

Question: Prove $3n^2 - 4n + 1 = \Theta(n^2)$ [Theeta]

Solution:

To prove $3n^2 - 4n + 1 = \Theta(n^2)$

We have to find c_1, c_2 and n_0 such that

$0 \le c_1n^2 \le 3n^2 - 4n + 1 \le c_2n^2 \qquad \forall\ n \ge n_0 \qquad (1)$

Let $c_1 = 2$, then we need n_0 for which above is true
We test it for $n_0 = 1, 2, 3$ and 4
For $n_0 = 4$, it is $32 \le 33$
Hence, $c_1 = 2$ for $\forall\ n \ge 4$

Now, let $c_2 = 3$, then we need n_0 for which (1) is true
Testing for value of n **from 1 onwards** in (1), we find $n_0 = 1$ as
$3(1)^2 - 4(1) + 1 \le 3(1)^2$
$0 \le 3$
Hence, $c_2 = 3$ for $\forall\ n \ge 1$

Finally, we have
$0 \le 2n^2 \le 3n^2 - 4n + 1 \le 3n^2 \qquad \forall\ n \ge 4$

Question: Find a sequence that satisfies the recurrence relation

$a_k = a_{k-1} + 2a_{k-2}$ $\forall k \geq 2$ and that also satisfies the initial condition $a_0 = 1$ and $a_1 = 8$

Solution:

The characteristic equation of the relation $a_k = a_{k-1} + 2a_{k-2}$ $\forall k \geq 2$ is

$t^2 - t - 2 = 0$

$\Rightarrow (t - 2)(t + 1) = 0$

$\Rightarrow t = -1, 2$

$r_n = 2^0, 2^1, 2^2, \ldots$ and $s_n = (-1)^0, (-1)^1, (-1)^2, \ldots$

Are both sequences which satisfy above relation but neither one is having $a_0 = 1$ and $a_1 = 8$

Now since $a_n = Cr_n + Ds_n$ also satisfies the same recurrence relation and

For n = 0 we get $a_o = Cr_o + Ds_o \Rightarrow 1 = C + D$ (1)

For n = 1 we get $a_1 = Cr_1 + Ds_1 \Rightarrow 8 = 2C - D$ (2)

Solving equation (1) and (2) we get, $C = 3$ *and* D = -2

Now $a_n = Cr_n + Ds_n \Rightarrow a_n = 3 \cdot (2)^n - 2(-1)^n$

is the required sequence which satisfies the given conditions

Theorem: Statement: Prove that any linear combination of solutions of equation given below is also a solution.

$$a_0t_n + a_1t_{n-1} + \ldots + a_kt_{n-k} = 0$$

Proof: Assume that f_n and g_n are solutions of above equation and hence satisfy it, i.e.

$$\sum_{i=0}^{k} a_i f_{n-i} = 0 \text{ and } \sum_{i=0}^{k} a_i g_{n-i} = 0$$

If we set $t_n = c.f_n + d.g_n$ for arbitrary constants c and d, we have to prove that t_n is also solution, i.e., $a_0t_n + a_1t_{n-1} + \ldots + a_kt_{n-k} = 0$

$$a_0t_n + a_1t_{n-1} + \ldots + a_k\ t_{n-k} =$$

$$a_0\ (cf_n + dg_n) + a1(cf_{n-1} + dg_{n-1}) + \ldots + a_k(cf_{n-k} + dg_{n-k}) =$$

$$c(\ a_0f_n + a_1f_{n-1} + \ldots + a_kf_{n-k}\ ) + d(a_0g_n + a_1g_{n-1} + \ldots + a_kg_{n-k}) =$$

$$= c.0 + d.0 = 0$$

Hence $a_0t_n + a_1t_{n-1} + \ldots + a_kt_{n-k} = 0$

Question: What is the solution of the recurrence relation?

$$a_n = a_{n-1} + 2a_{n-2} \quad \text{with } a_0=2 \text{ and } a_1=7?$$

Solution:

Since it is linear homogeneous recurrence, first find its characteristic equation

$r^2 - r - 2 = 0$

$(r+1)(r-2) = 0$

$r_1 = 2$ and $r_2 = -1$

So, by theorem $a_n = \propto_1 2^n + \propto_2 (-1)^n$ is a solution.

Now we should find $\propto_1$ and $\propto_2$ using initial conditions.

$a_0 = \propto_1 + \propto_2 = 2$

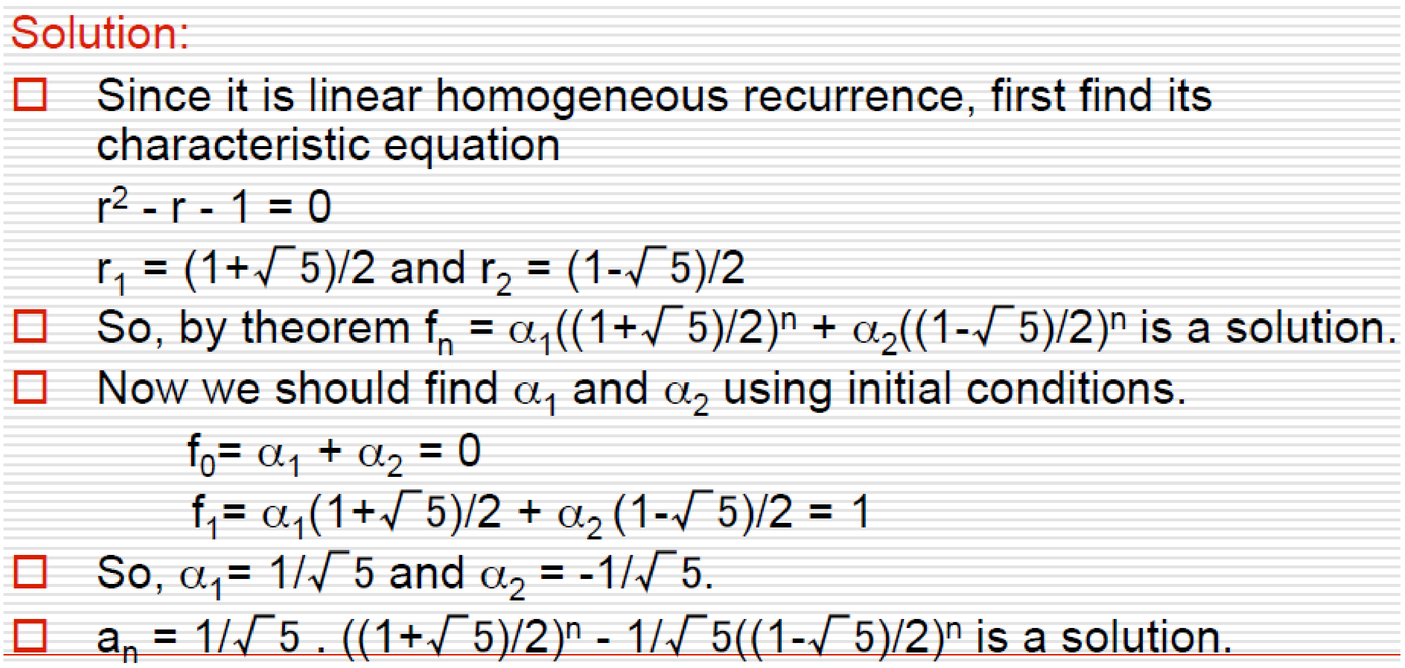
$a_1 = \propto_1 2 + \propto_2 (-1) = 7$

So, $\propto_1 = 3$ and $\propto_2 = -1$.

$a_n = 3 \,.\, 2^n - (-1)^n$ is a solution.

Question: What is the solution of the recurrence relation?

$f_n = f_{n-1} + f_{n-2}$ with $f_0=0$ and $f_1=1$?

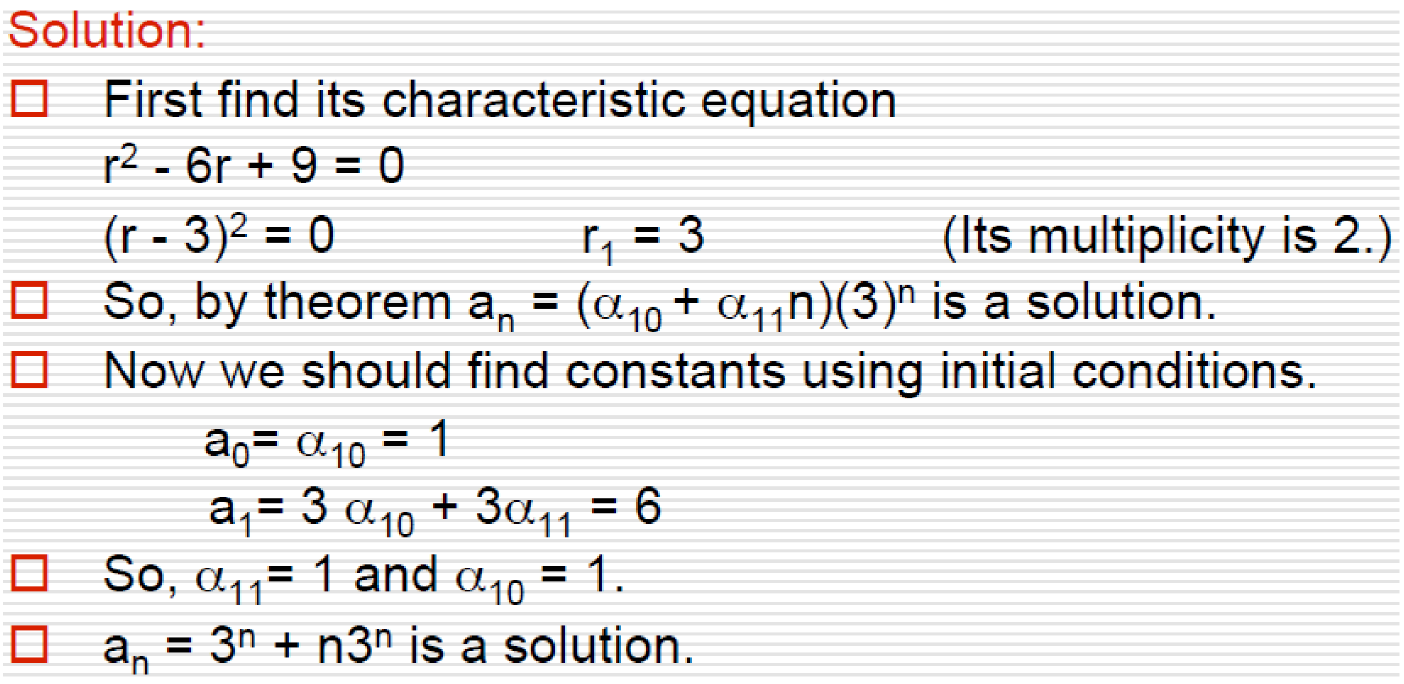


Solution:

- Since it is linear homogeneous recurrence, first find its characteristic equation
  $r^2 - r - 1 = 0$
  $r_1 = (1+\sqrt{5})/2$ and $r_2 = (1-\sqrt{5})/2$
- So, by theorem $f_n = \alpha_1((1+\sqrt{5})/2)^n + \alpha_2((1-\sqrt{5})/2)^n$ is a solution.
- Now we should find α_1 and α_2 using initial conditions.
  $f_0 = \alpha_1 + \alpha_2 = 0$
  $f_1 = \alpha_1(1+\sqrt{5})/2 + \alpha_2(1-\sqrt{5})/2 = 1$
- So, $\alpha_1 = 1/\sqrt{5}$ and $\alpha_2 = -1/\sqrt{5}$.
- $a_n = 1/\sqrt{5} . ((1+\sqrt{5})/2)^n - 1/\sqrt{5}((1-\sqrt{5})/2)^n$ is a solution.

Question: What is the solution of the recurrence relation?

$a_n = 6a_{n-1} - 9a_{n-2}$ with $a_0=1$ and $a_1=6$?



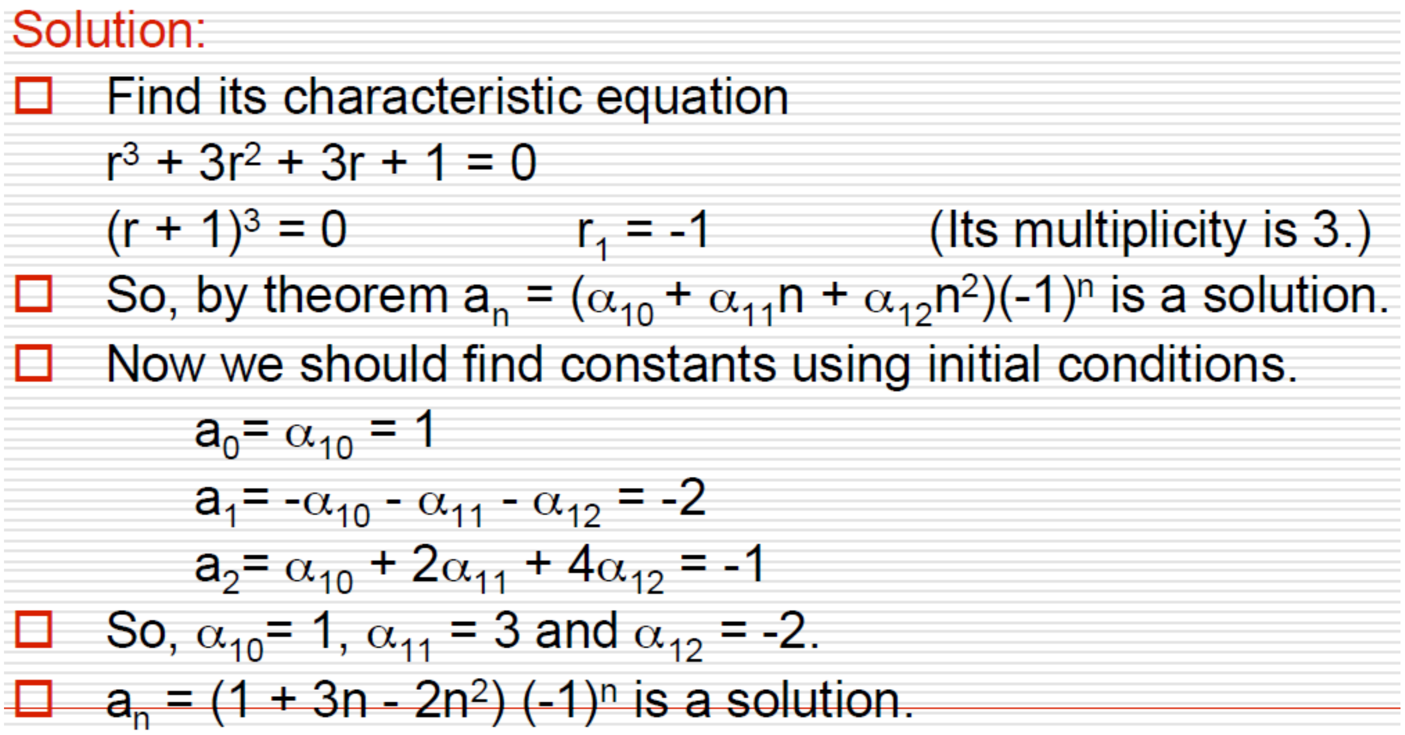
Solution:

- First find its characteristic equation
  $r^2 - 6r + 9 = 0$
  $(r - 3)^2 = 0$ $r_1 = 3$ (Its multiplicity is 2.)
- So, by theorem $a_n = (\alpha_{10} + \alpha_{11}n)(3)^n$ is a solution.
- Now we should find constants using initial conditions.
  $a_0 = \alpha_{10} = 1$
  $a_1 = 3\,\alpha_{10} + 3\alpha_{11} = 6$
- So, $\alpha_{11} = 1$ and $\alpha_{10} = 1$.
- $a_n = 3^n + n3^n$ is a solution.

Question: What is the solution of the recurrence relation?

$$a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$$

with $a_0=1$, $a_1=-2$ and $a_2=-1$?



Solution:

- Find its characteristic equation

  $r^3 + 3r^2 + 3r + 1 = 0$

  $(r + 1)^3 = 0$ $r_1 = -1$ (Its multiplicity is 3.)
- So, by theorem $a_n = (\alpha_{10} + \alpha_{11}n + \alpha_{12}n^2)(-1)^n$ is a solution.
- Now we should find constants using initial conditions.

  $a_0 = \alpha_{10} = 1$

  $a_1 = -\alpha_{10} - \alpha_{11} - \alpha_{12} = -2$

  $a_2 = \alpha_{10} + 2\alpha_{11} + 4\alpha_{12} = -1$
- So, $\alpha_{10} = 1$, $\alpha_{11} = 3$ and $\alpha_{12} = -2$.
- $a_n = (1 + 3n - 2n^2)(-1)^n$ is a solution.

Question: Show that f(n) = $n^2 + 2n + 1$ is $O(n^2)$.

Choose $k = 1$.

Assuming $n > 1$, then

$$\frac{f(n)}{g(n)} = \frac{n^2 + 2n + 1}{n^2} < \frac{n^2 + 2n^2 + n^2}{n^2} = 4$$

Choose $C = 4$. Note that $2n < 2n^2$ and $1 < n^2$.

Thus, $n^2 + 2n + 1$ is $O(n^2)$ because $n^2 + 2n + 1 \leq 4n^2$ whenever $n > 1$.

Question: Show that f(n) = 3n + 7 is O(n)

Choose $k = 1$.

Assuming $n > 1$, then

$$\frac{f(n)}{g(n)} = \frac{3n + 7}{n} < \frac{3n + 7n}{n} = \frac{10n}{n} = 10$$

Choose $C = 10$. Note that $7 < 7n$.

Thus, $3n + 7$ is $O(n)$ because $3n + 7 \leq 10n$ whenever $n > 1$.

Question: Show that f(n) = $(n + 1)^3$ is $O(n^3)$

Choose $k = 1$.

Assuming $n > 1$, then

$$\frac{f(n)}{g(n)} = \frac{(n+1)^3}{n^3} < \frac{(n+n)^3}{n^3} = \frac{8n^3}{n^3} = 8$$

Choose $C = 8$. Note that $n + 1 < n + n$ and $(n+n)^3 = (2n)^3 = 8n^3$. Thus, $(n+1)^3$ is $O(n^3)$ because $(n + 1)^3 \leq 8n^3$ whenever $n > 1$.

Question: Show that f(n) = $\sum_{i=1}^{n} i$ is $O(n^2)$

Choose $k = 1$.

Assuming $n > 1$, then

$$\frac{f(n)}{g(n)} = \frac{\Sigma_{i=1}^{n} i}{n^2} \leq \frac{\Sigma_{i=1}^{n} n}{n^2} = \frac{n^2}{n^2} = 1$$

Choose $C = 1$. Note that $i \leq n$ because n is the upper limit. Thus, $\Sigma_{i=1}^{n} i$ is $O(n^2)$ because $\Sigma_{i=1}^{n} i \leq n^2$ whenever $n > 1$.

Question: Prove $n^2 \log n = \Theta(n^2)$ is incorrect. [Theeta]

Proof: In order to prove $n^2 \log n = \Theta(n^2)$ is incorrect, we need to prove $n^2 \log n = O(n^2)$ is incorrect.

Let us prove $n^2 \log n = O(n^2)$ incorrect by contradiction.

Assume $n^2 \log n = O(n^2)$, that is, there exist positive constants c and n_0 such that $n^2 \log n <= cn^2$ for all $n >= n_0$ .... (1)

So $\log n <= c$ (by dividing n^2 to both sides of (1), and $n^2 > 0$)

But we know that $\lim\limits_{n \to \infty} \log n = \infty$, so we can't find such a constant c that make $\log n <= c$, we meet a contradiction.

Therefore, $n^2 \log n = O(n^2)$ is incorrect.

Then we proved $n^2 \log n = \Theta(n^2)$ is incorrect.

Question: To Prove n log n ϵ O(n^2)

Proof:

By the definition of big-O, we must find values of c and n_0 such that for all $n \geq n_0$, $n \log n \leq cn^2$.

Consider c=1 and $n_0 = 1$.

For all $n \geq 1$, $\log n \leq n$.

Therefore, $\log n \leq cn$, since c=1.

Multiplying both sides by n (and since $n \geq n_0 = 1$), we have $n \log n \leq cn^2$.

Therefore, $n \log n \in O(n^2)$.

Question: Prove $n^2 + 42n + 7 = O(n^2)$

Solution

Prove $n^2 + 42n + 7 = O(n^2)$

$$\begin{aligned} n^2 + 42n + 7 &\leq n^2 + 42n^2 + 7n^2 \quad \text{for } n \geq 1 \\ &= 50n^2 \end{aligned}$$

So, $n^2 + 42n + 7 \leq 50n^2$ for all $n \geq 1$

$n^2 + 42n^2 + 7n^2 = O(n^2)$ [$c = 50$, $n_0 = 1$]

Question: Show that (p ∨ q) ∧ (¬ p ∨ r) → (q ∨ r) is a tautology?

p	q	r	p ∨ q	(¬ p	(¬ p ∨ r	(p ∨ q) ∧ (¬ p ∨ r	(q ∨ r)	
T	T	T	T	F	T	**T**	**T**	**T**
T	T	F	T	F	F	F	T	T
T	F	T	T	F	T	**T**	**t**	**T**
T	F	F	T	F	F	F	f	T
F	T	T	T	T	T	T	T	T
F	T	F	T	T	T	T	T	T
F	F	T	F	T	T	F	t	T
F	F	F	F	T	T	F	F	T

Designing Algorithms using Divide & Conquer Approach

Divide and Conquer Approach

A general Divide and Conquer Algorithm

Step 1: If the problem size is small, solve this problem directly otherwise, split the original problem into 2 or more sub-problems with almost equal sizes.

Step 2: Recursively solve these sub-problems by applying this algorithm.

Step 3: Merge the solutions of the sub- problems into a solution of the original problem.

Time Complexity of General Algorithms

Time Complexity: $$T(n) = \begin{cases} 2T(n/2) + S(n) + M(n), & n \geq c \\ b \quad , & n < c \end{cases}$$

where S(n) is time for splitting

M(n) is time for merging

B and c are constants

Examples are *binary search*, *quick sort* and *merge sort*.

Merge Sort:

Merge-sort is based on divide-and-conquer approach and can be described by the following three steps:

Divide Step:

- If given array A has zero or one element, return S.
- Otherwise, divide A into two arrays, A1 and A2,
- Each containing about half of the elements of A.

Recursion Step:

- Recursively sort array A1, A2

Conquer Step:

- Combine the elements back in A by merging the sorted arrays A1 and A2 into a sorted sequence.

Visualization of Merge-sort as Binary Tree

- We can visualize Merge-sort by means of binary tree where each node of the tree represents a recursive call
- Each external node represents individual elements of given array A.
- Such a tree is called Merge-sort tree.
- The heart of the Merge-sort algorithm is conquer step, which merge two sorted sequences into a single sorted sequence

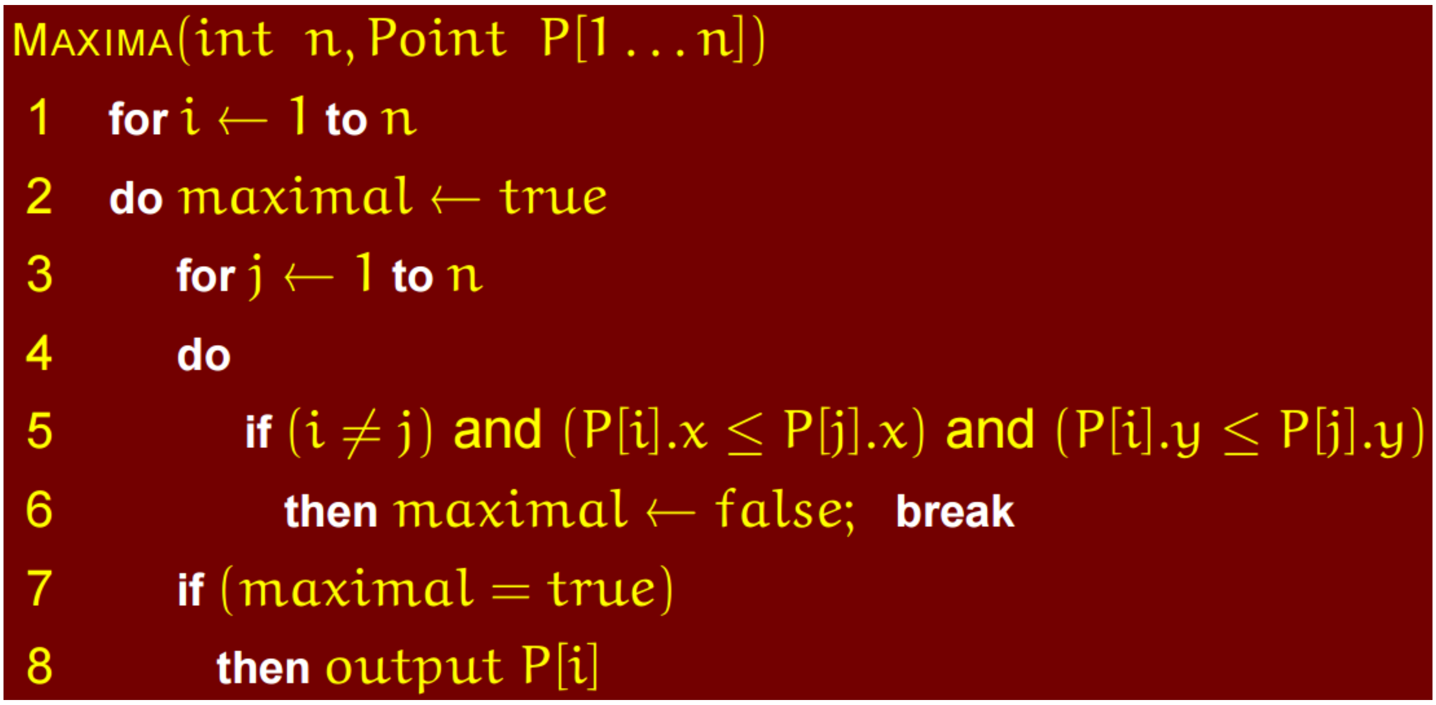
Question: Algorithm of matrix chain multiplication

Chain-Matrix-Order(p)

```
1.  n ← length[p] – 1
2.  for i ← 1 to n
3.      do m[i, i] ← 0
4.  for l ← 2 to n,
5.      do for i ← 1 to n-l+1
6.          do j ← i+l-1
7.              m[i, j] ← ∞
8.              for k ← i to j-1
9.                  do q ← m[i, k] + m[k+1, j]
                                + p_{i-1} . p_k . p_j
10.                     if q < m[i, j]
11.                         then m[i, j] = q
12.                             s[i, j] ← k
13. return m and s, “l is chain length”
```

Question: Algorithm of 2-Dimension points. [Maxima]

Solution:



```
MAXIMA($int\ n, Point\ P[1 \ldots n]$)
for $i \leftarrow 1$ to $n$
do $maximal \leftarrow true$
    for $j \leftarrow 1$ to $n$
    do
        if $(i \neq j)$ and $(P[i].x \leq P[j].x)$ and $(P[i].y \leq P[j].y)$
          then $maximal \leftarrow false$;  break
    if $(maximal = true)$
      then $output\ P[i]$
```

Question: The sequence <u$_n$> is defined by the recurrence [3u$_n$ + 1 / 5u$_n$ + 3]

$$u_{n+1} = \frac{3u_n+1}{5u_n+3}$$

You have initial conditionu1 = 1, Now you have to show *un* in terms of Fibonacci / Lucasnumbers.

Solution:

We first prove a straightforward lemma.
Lemma: For all n>=1;

$$3F_{2n-1} + L_{2n-1} = 2F_{2n+1}$$

$$5F_{2n-1} + 3L_{2n-1} = 2L_{2n+1} \qquad (1), (2)$$

Proof (of the Lemma): From straightforward manipulations of the Binet formulas for F$_n$ and L$_n$; one can easily show that

$$F_n + L_n = 2F_{2n+1} \qquad (3)$$

and

$$5F_n + L_n = 2L_{n+1} \qquad (4)$$

Thus we have

$$3F_{2n-1} + L_{2n-1} = 2F_{2n-1} + (F_{2n-1} + L_{2n-1})$$

$$= 2F_{2n-1} + 2F_{2n}$$

From (3)

$$= 2F_{2n+1},$$

Which is (1) also,

$$5F_{2n-1} + 3L_{2n-1} = (5F_{2n-1} + L_{2n-1}) + L_{2n-1}$$

$$= 2L_{2n} + 2L_{2n-1}$$

From (4)

$$= 2L_{2n+1},$$

Which is (2) also

Now we can prove the result in Question.

Theorem: For all n>=1,

$$v_n = \frac{F_{2n-1}}{L_{2n-1}}$$

Let $v_n = \frac{F_{2n-1}}{L_{2n-1}}$ so our goal is to prove that $v_n = u_n$ for all n>=1 it is clear that $v_1 = u_1$

Hence we need only check that $u_n$satisfies

$$u_{n+1} = \frac{3u_n + 1}{5u_n + 3}$$

Notice that

$$\frac{\mathbf{3u_n}+1}{\mathbf{5u_n}+3}=\frac{\mathbf{3}\frac{F_{2n-1}}{L_{2n-1}}+1}{\mathbf{5}\frac{F_{2n-1}}{L_{2n-1}}+3}$$

$$=\frac{\mathbf{3}F_{2n-1}+L_{2n-1}}{\mathbf{5}F_{2n-1}+3L_{2n-1}}$$

$$=\frac{2F_{2n+1}}{2L_{2n+1}}$$

From Lemma above

$$=\frac{F_{2n+1}}{L_{2n+1}}$$

$$=u_{n+1}$$

Question: Find a general formula for the Fibonacci sequence

$[f_n = f_{n-1} + f_{n-2}]$

$$\begin{cases} f_n & = & f_{n-1} + f_{n-2} \\ f_0 & = & 0 \\ f_1 & = & 1 \end{cases}$$

Solution. The characteristic equation $r^2 = r + 1$ has two distinct roots

$$r_1 = \frac{1+\sqrt{5}}{2} \quad \text{and} \quad r_2 = \frac{1-\sqrt{5}}{2}.$$

The general solution is given by

$$f_n = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Set

$$\begin{cases} 0 = c_1 + c_2 \\ 1 = c_1 \left(\frac{1+\sqrt{5}}{2}\right) + c_2 \left(\frac{1-\sqrt{5}}{2}\right). \end{cases}$$

We have $c_1 = -c_2 = \frac{1}{\sqrt{5}}$. Thus

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n, \quad n \geq 0.$$

Remark. The Fibonacci sequence f_n is an integer sequence, but it “looks like” a sequence of irrational numbers from its general formula above.

Question: Find the solution for the recurrence relation

[$x_n = 6x_{n-1} - 9x_{n-2}$]

$$\begin{cases} x_n &= 6x_{n-1} - 9x_{n-2} \\ x_0 &= 2 \\ x_1 &= 3 \end{cases}$$

Solution. The characteristic equation

$$r^2 - 6r + 9 = 0 \Longleftrightarrow (r-3)^2 = 0$$

has only one root $r = 3$. Then the general solution is

$$x_n = c_1 3^n + c_2 n 3^n.$$

The initial conditions $x_0 = 2$ and $x_1 = 3$ imply that $c_1 = 2$ and $c_2 = -1$. Thus the solution is

$$x_n = 2 \cdot 3^n - n \cdot 3^n = (2-n)3^n, \ \ n \geq 0.$$

Question: Find the solution for the recurrence relation

[$x_n = 2x_{n-1} - 5x_{n-2}$]

$$\begin{cases} x_n = 2x_{n-1} - 5x_{n-2}, & n \geq 2 \\ x_0 = 1 \\ x_1 = 5 \end{cases}$$

Solution. The characteristic equation

$$r^2 - 2r + 5 = 0 \Longleftrightarrow (x - 1 - 2i)(x - 1 + 2i) = 0$$

has two distinct complex roots $r_1 = 1+2i$ and $r_2 = 1-2i$. The initial conditions imply that

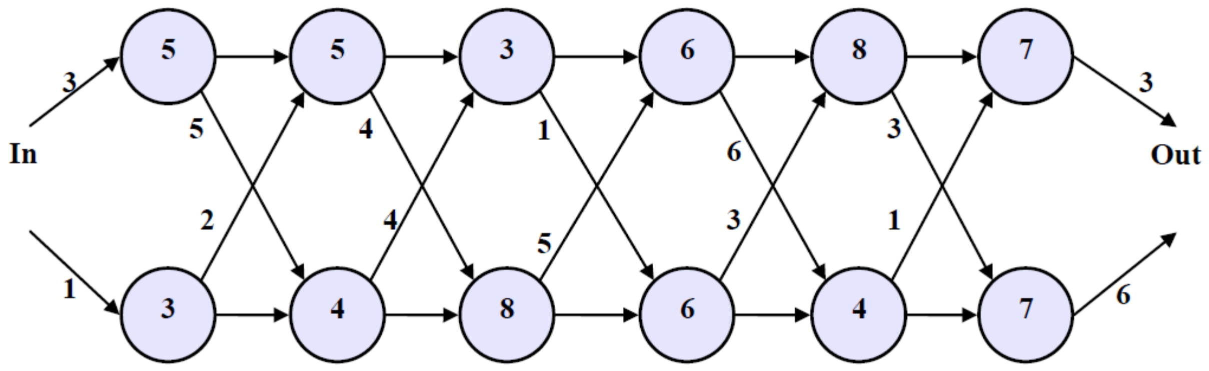
$$c_1 + c_2 = 1 \quad c_1(1+2i) + c_2(1-2i) = 5.$$

So $c_1 = \frac{1-2i}{2}$ and $c_2 = \frac{1+2i}{2}$. Thus the solutions is

$$\begin{aligned} x_n &= \frac{1-2i}{2} \cdot (1+2i)^n + \frac{1+2i}{2} \cdot (1-2i)^n \\ &= \frac{5}{2}(1+2i)^{n+1} + \frac{5}{2}(1-2i)^{n+1}, \ n \geq 0. \end{aligned}$$

Remark. The sequence is obviously a real sequence. However, its general formula involves complex numbers.

Question – 3: There are two assembly lines, as shown in the diagram below, each with 6 stations. The auto is required to go through from all of these 6 stations from left to right. Nodes represent stations. The assembly time at each station is shown at each node. The entering and exit times for an auto are also given. The transfer time is represented at the edges when an auto moves to next station on a different line. There is no transfer time if it stays on the same line. Determine which stations to choose from lines 1 and 2 to minimize total time through the factory. Also compute the optimal value in terms of time. Use Dynamic Programming Approach. You need to calculate $f_i[j]$, $l_i[j]$, f*, l* and the optimal path.



Solution:

Let

$f_i[j]$ = Denotes the fastest time from starting point station $S_{i,j}$

$f_1[j]$ = Denotes the fastest possible time from starting point through j^{th} station of assembly line 1.

$f_2[j]$ = Denotes the fastest possible time from starting point through j^{th} station of assembly line 2.

$l_i[j]$ = The line number, 1 or 2, whose station j-1 is used in a fastest way through station $S_{i,j}$.

It is to be noted that $l_i[1]$ is not required to be defined because there is no station before 1

$t_i[j-1]$ = Denotes the transfer time cost from assembly line i to station $S_{i-1,j}$ or $S_{i+1,j}$

Objective function f* = $\min(f_1[n] + x_1, f_2[n] + x_2)$

l* = Denotes the line number whose last or n^{th} station is used in a fastest way through entire factory.

Mathematical Model: Finding Objective Function

$f_1[1] = e_1 + a_{1,1}$

$f_2[1] = e_2 + a_{2,1}$

$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$ for $j \geq 2$;

$f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$ for $j \geq 2$;

Base Cases

$f_1[1] = e_1 + a_{1,1}$	$f_1[1] = 3 + 5$	$\mathbf{f_1[1] = 8}$
$f_2[1] = e_2 + a_{2,1}$	$f_2[1] = 1 + 3$	$\mathbf{f_2[1] = 4}$

Computation of $f_1[2]$, i.e. $j = 2$

$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$

$f_1[2] = \min(f_1[1] + a_{1,2}, f_2[1] + t_{2,1} + a_{1,2})$

$f_1[2] = \min(8 + 5, 4 + 2 + 5) = \min(13,11)$

$\mathbf{f_1[2] = 11}$ $\mathbf{l_1[2] = 2}$

Computation of $f_2[2]$, i.e. $j = 2$

$f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$

$f_2[2] = \min(f_2[1] + a_{2,2}, f_1[1] + t_{1,1} + a_{2,2})$

$f_2[2] = \min(4 + 4, 8 + 5 + 4) = \min(8,17)$

$\mathbf{f_2[2] = 8}$ $\mathbf{l_2[2] = 2}$

Computation of $f_1[3]$, i.e. $j = 3$

$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$

$f_1[3] = \min (f_1[2] + a_{1,3} , f_2[2] + t_{2,2} + a_{1,3})$

$f_1[3] = \min (11 + 3 , 8 + 4 + 3) = \min (14, 15)$

$\mathbf{f_1[3] = 14} \qquad \mathbf{l_1[3] = 1}$

Computation of $f_2[3]$, i.e. j = 3

$f_2[j] = \min (f_2[j-1] + a_{2,j} , f_1[j-1] + t_{1,j-1} + a_{2,j})$

$f_2[3] = \min (f_2[2] + a_{2,3} , f_1[2] + t_{1,2} + a_{2,3})$

$f_2[3] = \min (8 + 8 , 11 + 4 + 8)$

$f_2[3] = \min (16 , 23)$

$\mathbf{f_2[3] = 16} \qquad \mathbf{l_2[3] = 2}$

Computation of $f_1[4]$, i.e. j = 4

$f_1[j] = \min (f_1[j-1] + a_{1,j} , f_2[j-1] + t_{2,j-1} + a_{1,j})$

$f_1[4] = \min (f_1[3] + a_{1,4} , f_2[3] + t_{2,3} + a_{1,4})$

$f_1[4] = \min (14 + 6 , 16 + 5 + 6)$

$f_1[4] = \min (20 , 27)$

$\mathbf{f_1[4] = 20} \qquad \mathbf{l_1[4] = 1}$

Computation of $f_2[4]$, i.e. j = 4

$f_2[j] = \min (f_2[j-1] + a_{2,j} , f_1[j-1] + t_{1,j-1} + a_{2,j})$

$f_2[4] = \min (f_2[3] + a_{2,4} , f_1[3] + t_{1,3} + a_{2,4})$

$f_2[4] = \min (16 + 6 , 14 + 1 + 6)$

$f_2[4] = \min (22 , 21)$

$\mathbf{f_2[4] = 21} \qquad \mathbf{l_2[4] = 1}$

Computation of $f_1[5]$, i.e. j = 5

$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$

$f_1[5] = \min(f_1[4] + a_{1,5}, f_2[4] + t_{2,4} + a_{1,5})$

$f_1[5] = \min(20 + 8, 21 + 3 + 8)$

$f_1[5] = \min(28, 32)$

$f_1[5] = 28$ $l_1[5] = 1$

Computation of $f_2[5]$, i.e. j = 5

$f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$

$f_2[5] = \min(f_2[4] + a_{2,5}, f_1[4] + t_{1,4} + a_{2,5})$

$f_2[5] = \min(21 + 4, 20 + 6 + 4)$

$f_2[5] = \min(25, 30)$

$f_2[5] = 25$ $l_2[5] = 2$

Computation of $f_1[6]$, i.e. j = 6

$f_1[j] = \min(f_1[j-1] + a_{1,j}, f_2[j-1] + t_{2,j-1} + a_{1,j})$

$f_1[6] = \min(f_1[5] + a_{1,6}, f_2[5] + t_{2,5} + a_{1,6})$

$f_1[6] = \min(28 + 7, 25 + 1 + 7)$

$f_1[6] = \min(35, 33)$

$f_1[6] = 33$ $l_1[6] = 2$

Computation of $f_2[6]$, i.e. j = 6

$f_2[j] = \min(f_2[j-1] + a_{2,j}, f_1[j-1] + t_{1,j-1} + a_{2,j})$

$f_2[6] = \min(f_2[5] + a_{2,6}, f_1[5] + t_{1,5} + a_{2,6})$

$f_2[6] = \min(25 + 7, 28 + 3 + 7)$

$f_2[6] = \min(32, 38)$

$\mathbf{f_2[6] = 32}$ $\mathbf{l_2[6] = 2}$

j	1	2	3	4	5	6
$f_1[j]$	8	11	14	20	28	33
$f_2[j]$	4	8	16	21	25	32

j	2	3	4	5	6
$l_1[j]$	2	1	1	1	2
$l_2[j]$	2	2	1	2	2

Keeping Track for Constructing Optimal Solution

$f^* = \min(f_1[6] + x_1, f_2[6] + x_2)$

$f^* = \min(33 + 3, 32 + 6)$

$f^* = \min(36, 38)$

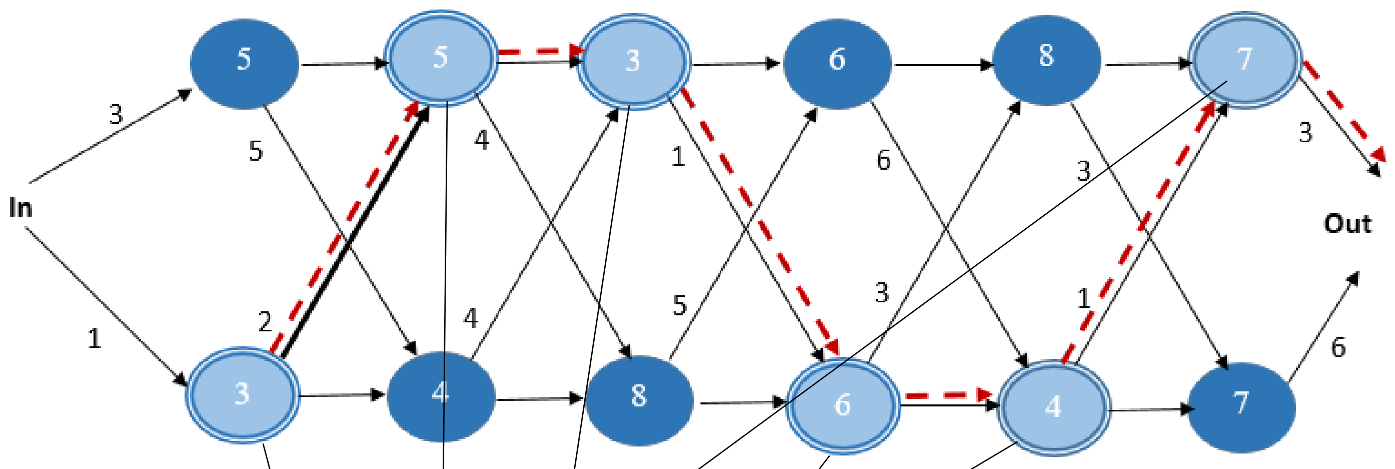
$\mathbf{f^* = 36}$

Hence, the minimum time required by auto from starting point to exit point of the factory is 36.

The auto will take minimum time from starting point to the exit point if and only if it has come out from the 6th station of assembly line 1.

Therefore, **l* = 1**

Fastest Way: Assembly-Line Scheduling – Optimal Path



The optimal path is shown by arrows

$l^* = 1$ => Station $S_{1,6}$

$l_1[6] = 2$ => Station $S_{2,5}$

$l_2[5] = 2$ => Station $S_{2,4}$

$l_2[4] = 1$ => Station $S_{1,3}$

$l_1[3] = 1$ => Station $S_{1,2}$

$l_1[2] = 2$ => Station $S_{2,1}$